Analiza poteka markovskih procesov pri modeliranju tekem v športnih igrah

Petar Vračar

doktorska disertacija

predana

Fakulteti za računalništvo in informatiko

kot del izpolnjevanja pogojev za pridobitev naziva

doktor znanosti

s področja

računalništva in informatike

Ljubljana, 2017

IZJAVA

Izjavljam, da sem avtor dela in da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali na drugem visokošolskem zavodu, razen v primerih, kjer so navedeni viri.

— Petar Vračar —

marec 2017

Oddajo so odobrili

dr. Igor Kononenko

redni profesor za računalništvo in informatiko

MENTOR IN ČLAN OCENJEVALNE KOMISIJE

dr. Marko Robnik Šikonja

izredni profesor za računalništvo in informatiko

PREDSEDNIK OCENJEVALNE KOMISIJE

dr. Frane Erčulj

redni profesor za področje znanost o športu - kineziologija

ZUNANJI ČLAN OCENJEVALNE KOMISIJE

Univerza v Ljubljani, Fakulteta za šport

PREDHODNA OBJAVA

Izjavljam, da so bili rezultati obravnavane raziskave predhodno objavljeni/sprejeti za objavo v recenzirani reviji ali javno predstavljeni v naslednjih primerih:

[1] Petar Vračar, Erik Štrumbelj, in Igor Kononenko. Modeling basketball play-by-play data. *Expert Systems with Applications*, 44:58–66, 2016.

Potrjujem, da sem pridobil pisna dovoljenja vseh lastnikov avtorskih pravic, ki mi dovoljujejo vključitev zgoraj navedenega materiala v pričujočo disertacijo. Potrjujem, da zgoraj navedeni material opisuje rezultate raziskav, izvedenih v času mojega podiplomskega študija na Univerzi v Ljubljani.

Mojemu očetu

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*
Petar Vračar
Analiza poteka markovskih procesov pri modeliranju tekem v športnih igrah

POVZETEK

Naraščujoče računske zmogljivosti in avtomatsko zbiranje čedalje bogatejših podatkov o poteku športnih dogodkov so omogočili razvoj kompleksnih analitičnih modelov, ki nosilcem odločanja ponujajo kompetitivno prednost v svetu športa. V disertaciji naslavljamo problem samodejnega izluščanja zakonitosti o sosledju dogodkov na športnih tekmah in gradnje statističnega modela za generiranje verodostojnih simulacij tekem med specificiranima ekipama. Razvoj športne tekme modeliramo kot slučajni sprehod v prostoru stanj. Matriko prehoda markovskega modela izražamo v funkcijski odvisnosti od opisa trenutnega stanja, ki vključuje faktorje, relevantne za nadaljnji potek dogodkov na tekmi. Osnovna ideja našega pristopa temelji na uporabi kaskade modelov, ki zaporedno (in pogojeno eden na drugega) napovedujejo posamezne dele opisa naslednjega stanja. Predstavljamo postopek za avtomatsko generiranje atributnega prostora, ki ne potrebuje domenskega predznanja. Atribute definiramo v obliki razmerja med številom vstopov in izstopov iz višjenivojskih konceptov, ki jih identificiramo kot množice podobnih dogodkov. Podobnost med dogodki ugotavljamo na podlagi podobnosti porazdelitev, ki opisujejo predhodne oziroma naslednje dogodke v opazovanih sekvencah razvoja športnih tekem. Eksperimentalna evalvacija predlaganih metod na košarkarski domeni je pokazala, da so modeli, dobljeni z učenjem na podlagi avtomatsko generiranih atributov, po kvaliteti napovedovanja naslednjega dogodka in časa med dogodki primerljivi z modeli, naučenimi z ekspertnimi atributi. Statistična analiza dobljenih simulacij je pokazala, da so modeli uspešno zajeli dinamiko razvoja košarkarske tekme.

Ključne besede: modeliranje športa, športne napovedi, markovski proces, konstrukcija atributov, simuliranje tekem

University *of Ljubljana*
Faculty *of Computer and Information Science*
Petar Vračar
Analysis of Markov processes in team sports modeling

ABSTRACT

The increasing computational power and automated collection of ever richer data about sporting events have enabled the development of complex analytical models that offer decision makers a competitive advantage in the world of sport. In the thesis we address the problem of automatic extraction of regularities in the sequence of events in sports games and construction of statistical models for generating a plausible simulation of a match between two distinct teams. We model the progression of a sports game as a random walk through the state space. We express the transition matrix of our Markov model as a function of the current state description which includes factors relevant for the further development of events in the match. The main idea of our approach is to incorporate a cascade of models that sequentially (and conditioned to each other) predict the individual components of the next state description. We present a method for automatic construction of a feature space which does not require any expert knowledge about the domain. The attributes are defined as the ratio between the number of entries and exits from higher-level concepts that are identified as groups of similar game events. The similarity between the events is determined by the similarity between probability distributions describing the preceding and following events in the observed sequences of game progression. Experimental evaluation of the proposed methods applied in the basketball domain showed that the models fitted in the automatically generated feature space are of comparable quality to models that use features based on expert knowledge. Statistical analysis of the generated simulations showed that the models successfully capture the dynamics of the game of basketball.

Key words: sports modelling, sports forecasting, Markov process, feature construction, match simulation

ZAHVALA

Zahvaljujem se mentorju Igorju za nepresušno zakladnico nasvetov - strokovnih in življenskih, ki jih je ponujal vsakič, ko sem jih potreboval.

Hvala Eriku in Mihi za njuno pripravljenost poslušati me in mi priskočiti na pomoč.

Odlika dobrih ekip je, da so več kot vsota posameznih članov. Menim, da je Laboratorij za kognitivno modeliranje ena takih ekip. Zahvaljujem se vsem članom laboratorija - z vašo pomočjo sem prevladal ovire, na katerih bi sam že zdavnaj padel.

Zahvaljujem se moji mami na brezmejni in brezpogojni podpori in ljubezni, ki mi ju daje celo življenje.

Hvala Nadi, ki v skladu s svojim imenom prinaša upanje in spokoj v moje življenje.

Hvala Nikoli, ki mi je podaril najpomembenjši naziv - tata.

— Petar Vračar, Ljubljana, marec 2017.

KAZALO

I

Uvod

Športna analitika je področje statistične obdelave podatkov, ki z analizo doseženih rezultatov in s projekcijo bodoče uspešnosti poskuša nosilcem odločanja ponuditi kompetitivno prednost v svetu športa. S svojimi začetki v 70-ih letih dvajsetega stoletja, ko so športni zanesenjaki iz skupine SABR (Society for American Baseball Research[1]) pričeli z izumljanjem inovativnih kazalcev uspešnosti igralcev bejzbola (t.i. "sabermetrična" revolucija [1]), je področje športne analitike svoj razcvet dobilo ob prelomu tisočletja. Naraščujoče računske zmogljivosti in avtomatsko zbiranje čedalje bogatejših podatkov o poteku športnih dogodkov so omogočili razvoj kompleksnejših analitičnih modelov. Športna analitika je dandanes prisotna praktično v vseh večjih športnih panogah in je nepogrešljivi del najuspešnejših športnih organizacij.

Količina podatkov, ki se zbira o poteku športnih dogodkov, je zdavnaj presegla mejo obvladljivosti z ročno obdelavo in je avtomatizirana analitika podatkov edina perspektivna pot. Metode podatkovnega rudarjenja in strojnega učenja [2] se uporabljajo na različnih področjih športne analitike - od napovedovanja izida bodočih tekem, preko kvantitativnega ocenjevanja zmogljivosti ekip in posameznikov, do odkrivanja novih zakonitosti in prvin športa. Pregled metod podatkovnega rudarjenja v športu najdemo v [3]. Vse te metode iz različnih zornih kotov prispevajo svoj del odgovora na ključno vprašanje - na kakšen način lahko ekipa (ali posameznik, če govorimo o individualnem športu) poveča možnost svoje zmage.

V disertaciji se ukvarajamo predvsem s praktičnim problemom, kako iz kronoloških zapisov o poteku dogajanja na športnih tekmah samodejno izluščiti interpretabilne zakonitosti sosledja dogodkov in na podlagi le-teh izdelati statistični model za generiranje verodostojnih simulacij tekem med specificiranima ekipama.

Generirane simulacije morajo v prvi vrsti biti skladne s pravili igre, ki jo modeliramo. Zajemati morajo tudi značilne vzorce poteka dogodkov, ki sledijo iz ustaljenih taktičnih prijemov in specifike trenutne situacije na tekmi (na primer, ekipa v rezultatskem zaostanku dvigne tempo igre, postane bolj agresivna in prevzema več tveganja). Nenazadnje, generirane simulacije morajo odražati tudi značilnosti ekip, ki igrajo simulirane tekme (na primer, če je košarkarska ekipa znana po dobrih strelcih, pričakujemo, da v večini simulacij izkažejo visok procent uspešnih metov iz igre). Drugače povedano, verodostojne simulacije se morajo po statističnih kazalcih igre v čim večji meri ujemati z dejanskimi tekmami.

Razvoj dogodkov na športni tekmi ni popolnoma naključen, temveč ga oblikujejo

[1] http://sabr.org/

zakonitosti igre in zmogljivosti igralcev. Nekateri izmed teh faktorjev vpliva so objektivno določljivi in so največkrat tudi eksplicitno podani v vhodnih podatkih (na primer: tip dogodka, igralni čas, trenutni rezultat in podobno). Drugi fatorji so subjektivni in so običajno posredno določeni na podlagi ekspertnega predznanja (na primer: zmogljivosti ekip ocenimo na podlagi doseženih statistik v prej odigranih tekmah). V kontekstu strojnega učenja predstavljajo faktorji vpliva atributni prostor, v katerem modeliramo razvoj dogodkov na tekmi. Učenje modela predstavlja oblikovanje funkcijske odvisnosti med verjetnostjo nastopa posameznega dogodka (odvisna spremenljivka) in trenutnimi vrednostmi faktorjev vpliva (neodvisne spremenljivke).

1.1 Prispevki k znanosti

V disertaciji podajamo dva prispevka k znanosti:

- *Postopek za samodejno oblikovanje atributnega prostora za modeliranje ravoja športnih tekem.* Postopek ne potrebuje domenskega predznanja in podaja interpretabilne definicije atributov. Uporabna vrednost predstavljenega postopka je dvojna: (1) omogoča modeliranje športov, kjer ni znana množica informativnih atributov in (2) interpretabilni atributi predstavljajo osnovo za nadaljno analizo obstoječih (in po možnosti tudi novoodkritih) konceptov modeliranega športa ter na ta način poglabljajo razumevanje vpliva posameznih faktorjev na potek igre.

- *Splošna metodologija za modeliranje razvoja športne tekme.* Metoda temelji na gradnji kaskade modelov, ki zaporedno napovedujejo relevantne vidike poteka dogodkov (na primer, pri modeliranju košarke ponavadi napovedujemo tip in čas nastopa naslednjega dogodka, pri nekaterih športih pa je potrebno napovedati tudi pozicijo na igrišču, kjer se bo zgodil ta dogodek, oziroma kakšen drug pomemben vidik simulacije). Vsak model podaja svoje napovedi v odvisnosti od trenutnega opisa tekme in pogojeno z napovedmi predhodnih modelov v kaskadi. Metodologija je uporabna pri modeliranju vseh športov, ki jih lahko dobro opišemo s stanji in prehodi med njimi. Verodostojen in natančen simulator športnih tekem je lahko sestavni del ekspertnega sistema, namenjenega trenerjem in strokovnemu osebju v klubih, ki bi ga uporabljali kot orodje za analizo, usmerjeno k dvigovanju kvalitete igre in izboljševanju doseženih rezultatov.

V drugem poglavju predstavimo pregled področja športne analitike s povdarkom na metodah za opisovanje zmogljivosti ekip, napovedovanje končnega izida in simuliranje

poteka športnih dogodkov. V tretjem poglavju podamo metodo za modeliranje razvoja športne tekme med specifičnima ekipama. Četrto poglavje opiše metodo za avtomatsko generiranje atributnega prostora. V petem poglavju predstavimo uporabo opisanih metod za modeliranje poteka košarkarske tekme in analiziramo verodostojnost dobljenih simulacij. Disertacijo zaključimo v šestem poglavju s predlogi za nadaljnje delo.

2

Pregled področja

Iz širokega razpona raziskav na področju športne analitike so za pričujoče delo najbolj relevantne metode za simuliranje poteka dogodkov na športni tekmi in v nekoliko širšem kontekstu tudi metode za napovedovanje izida športne tekme. Modeli za napovedovanje poteka dogodkov potrebujejo značilke za opis nasprotnikov, zato so relevantne tudi metode za modeliranje moči ekip oziroma opredelitev značilnosti njihove igre.

2.1 Ocenjevanje moči ekip

Športne ekipe lahko opišemo z oceno njihove kvalitete. Rangirna metoda ELO [4], originalno razvita za primerjanje moči šahovskih igralcev, pozneje prirejena tudi za ekipne športe ([5], [6], [7], [8]), iterativno posodablja ocene moči ekip po vsaki odigrani tekmi. Na začetku se vsem ekipam določi povprečna moč, ki se potem popravlja glede na razliko med dejanskim izidom tekme in izidom, ki bi ga pričakovali glede na ocenjeno moč tekmecev. Glavna pomanjkljivost metode ELO je neupoštevanje stopnje zanesljivosti ocen tekmecev, ko so le-te določene na podlagi relativno malo in po možnosti tudi časovno zelo razmaknjenih tekem. Posplošitve metode ELO, kot so sistemi Glicko [9], Glicko-2 [10] in Trueskill [11], rešujejo problem nezanesljivosti ocen tako, da moči ekip ne povzemajo z eno samo vrednostjo, temveč jih obravnavajo kot naključne spremenljivke s podanim povprečjem in varianco (drugače povedano, moč ekipe modelirajo kot parametrično verjetnostno porazdelitev).

Ocenjevanje moči ekip po principu najmanjših kvadratov (ang. least squares) je optimizacijski problem na usmerjenem grafu, kjer vozlišča predstavljajo ekipe, povezave med vozlišči (njihova smer in utež) pa ustrezajo empirični parni primerjavi med ekipami. Naloga postopka je določiti vrednosti v vozliščih (moči ekip) tako, da se čim bolj ujemajo z empiričnimi podatki v povezavah. Prvi je v športnem kontektu omenjeno metodo uporabil Leake [12], sledile pa so izpeljanke ([13], [14]), ki se razlikujejo v načinu, kako iz dejanskih tekem določijo uteži na povezavah.

Keener [15] je predlagal postopek za konstrukcijo absolutne moči ekip na podlagi njihovih relativnih moči. Slednje so organizirane v obliki kvadratne matrike z elementi, ki ustrezajo neposredni primerjavi dveh ekip in sledijo iz rezultatov odigranih tekem. Absolutne moči ekip ustrezajo dominantnemu lastnemu vektorju matrike relativnih moči.

Kvam in Sokol [16] sta za ocenjevanje moči ekip uporabila markovsko verigo, kjer vsako stanje predstavlja posamezno ekipo. Prehod med stanji ponazarja glas hipotetičnega volivca, ki se odloča o tem, katera ekipa je najmočnejša. V vsakem koraku volivec

naključno izbere odigrano tekmo ekipe, za katero trenutno meni, da je najboljša, ter se v skladu z rezultatom tekme z določeno verjetnostjo odloči podati glas zmagovalcu oziroma poraženca. Stacionarno stanje te markovske verige ustreza razvrstitvi ekip po moči. Za modeliranje verjetnosti prehodov je bil uporabljen logistični model. Predlagana metoda se je izkazala kot boljši napovedovalec razpleta ameriškega študentskega košarkarskega prvenstva NCAA od mnogih drugih statističnih modelov in anket. Brown in Sokol [17] sta opisano metodo dodatno izboljšala tako, da sta pri modeliranju prehodov uporabila bayesovski pristop.

Govan in Meyer [18] sta priredila algoritem PageRank za rangiranje ekip v ligi ameriškega nogometa NFL (National Football League). Ideja postopka je v tem, da je kvaliteta ekipe izražena kot utežena vsota kvalitete v medsebojnih tekmah premaganih nasprotnikov.

2.2 Ocenjevanje učinkovitosti igre

Opis zmogljivosti ekip lahko razširimo z značilkami učinkovitosti različnih aspektov njihove igre. Preproste značilke so podane v obliki osnovnih opisnih statistik, ki temeljijo na štetju ključnih dogodkov na odigranih tekmah. Z bogatenjem zapisov o poteku tekem so se razvile tudi bolj sofisticirane značilke učinkovitosti igre, ki so praviloma ročno določene na podlagi ekspertnega znanja o športu.

Košarka je bila zaradi svoje dinamike, popularnosti in nenazadnje dostopnosti statističnih podatkov deležna velike pozornosti raziskovalcev. Eden ključnih konceptov za analizo učinkovitosti igre je koncept posesti žoge. Posest se prične, ko ekipa pridobi popolno kontrolo žoge in se konča, ko žogo prevzame nasprotnik. Ker se ekipi izmenjujeta v posesti žoge, bo število posesti na koncu tekme približno enako za obe ekipi[1]. Iz tega sledi, da ločilnico med zmagovalci in poraženci predstavlja učinkovito izkoriščanje lastnih posesti (in onemogočanje nasprotnikovih posesti). Različni aspekti te učinkovitosti so tako osnova naprednejših opisnih statistik ([19], [20], [21], [22]). Izmed teh izpostavljamo t.i. štiri faktorje (ang. four factors), ki razčlenjujejo učinkovitost igre v napadu in obrambi s pomočjo štirih košarkarskih prvin: uspešnost meta iz igre, povprečno število izgubljenih žog na posest, delež dobljenih skokov in uspešnost izvajanja prostih metov. Raziskave so pokazale, da štirje faktorji močno korelirajo z deležem doseženih zmag ([23], [24]), primerni pa so tudi za opisovanje moči ekip pri

[1] Do razlik lahko pride glede na to, katera ekipa ima zadnjo posest v četrtini.

modeliranju poteka košarkarske tekme ([25], [26]).

Tradicionalni kazalci merijo uspešnost rezultata akcij (delež zadetih metov, odstotek dobljenih skokov in podobno), v novejšem času se več pozornosti posveča ocenjevanju procesa razvoja igre. V zadnjih letih so postali javno dostopni prostorsko-časovni podatki o gibanju žoge in igralcev med celotno tekmo, kar je raziskovalcem odprlo nove možnosti za razvoj značilk za opisovanje prej nedostopnih aspektov igre, kot so na primer dinamika gradnje napada, postavljanje obrambe, pokrivanje prostora in podobno. Za razliko od play-by-play podatkov v obliki nizanja samo končnih dogodkov, prostorsko-časovni podatki o gibanju igralcev omogočajo ključno novost - analizo celotnega poteka posesti žoge. Cervone in sod. [27] so predstavili mero za povzemanje napadalne faze igre v obliki krivulje EPV (ang. Expected Possession Value), ki na podlagi trenutne prostorske konfiguracije igralcev in žoge opredeljuje pričakovano število doseženih točk ob koncu posesti.

Franks in sod. [28] so prav tako uporabili prostorsko-časovne podatke za ocenjevanje učinkovitosti košarkarjev v fazi obrambe. S pomočjo skritega markovskega modela se najprej za vsakega obrambnega igralca ugotovi, koga je pokrival v danem trenutku, nato se ovrednoti dinamika in organizacija igre v obrambi v obliki entropije - negotovosti pri določanju zadolžitev med nasprotnikovim napadom. Mera se lahko uporabi za karakterizacijo obrambe pa tudi napada v smislu, koliko nereda povzročajo v nasprotnikovi obrambi.

Lucey in sod. [29] so modelirali verjetnost zadetka v nogometu. Analizirali so 9732 strelov na gol na podlagi 10 sekundnega časovnega okna s podatki o gibanju igralcev in žoge na igrišču. Uporabili so logistično regresijo za modeliranje uspešnosti strela na gol v odvisnosti od pozicije žoge, števila in razporeditve obrambnih igralcev, ter tipa akcije (običajni napad, protinapad, izvajanje kota, kazenski strel, prosti strel ali predložek). Na podlagi dobljenega modela so ocenili učinkovitost nogometnih moštev v napadalnih in obrambnih akcijah.

Obsežen pregled drugih raziskav na podlagi prostorsko-časovnih podatkov (razpoznavanje formacij in taktičnih prijemov, analiza vzorcev podaj, modeliranje gibanja igralcev in podobno) lahko najdemo v [30].

2.3 Modeliranje izida tekme

Statistični modeli za napovedovanje izida tekem so velikokrat zasnovani tako, da ustrezajo obliki napovedi, kot jih postavljajo stavnice. Za tekme ameriškega nogometa je

najpopularnejša oblika stav na razliko v končnem rezultatu, zato večina modelov napoveduje tovrstni rezultat.

Zuber in sod. [31] so modelirali razliko v rezultatu kot linearno kombinacijo razlik vrednosti opisnih statistik nasprotnikov, ki segajo od osvojenih jardov s tekom oziroma podajami, deleža poskusov podaj med vsemi poskusi, števila prestreženih podaj pa do števila novincev v ekipi in števila zmag. Vse opisne statistike so bile izračunane iz javno dostopnih podatkov.

Dana in Knetter [32] sta prav tako modelirala razliko v rezultatu v odvisnosti od lastne moči ekip in prednosti domačega igrišča (konstanta). Moč ekipe predstavlja razliko v zmogljivosti v napadu in obrambi, izražena pa je v številu točk. Empirično sta ovrednotila dve inačici modela - s statičnimi in dinamičnimi močmi ekip ter ju primerjala z napovedmi stavnic. Eksperimentalna evalvacija na podlagi 609 tekem iz sezon NFL 1980, 1981, 1988 in 1989 je pokazala, da so dobljene napovedi po kvaliteti primerljive z napovedmi stavnic, a so pod spodnjo mejo dobičkonostnosti.

Glickman in Stern [33] sta uporabila model v prostoru stanj (ang. state-space model) za modeliranje kratkoročnih (med igralnimi krogi znotraj sezone) in dolgoročnih (med sezonami) sprememb moči ekip ter prednosti domačega igrišča za vsako ekipo posebej. Model sta uporabila za napovedovanje razlike v rezultatu in na omejeni testni množici (110 tekem v drugi polovici sezone NFL 1993) premagala napovedi stavnic.

Sinha in sod. [34] so uporabili objave na družbenem omrežju Twitter kot dodaten vir informacij za napovedovanje izida tekem ameriškega nogometa. Zbrane objave so najprej razvrstili glede na omembo ekipe, nato pa še glede na čas objave (pred tekmo, neposredno pred začetkom tekme, po končani tekmi). Analizo objav so izvedli po modelu vreče besed (ang. bag-of-words) in uporabili unigrame kot preproste značilke. Eksperimentalna evalvacija na treh sezonah NFL (2010-2012) je pokazala, da značilke, pridobljene iz družbenih medijev, lahko izboljšajo kvaliteto napovedi modelov, ki uporabljajo le tradicionalne opisne statistike na podlagi odigranih tekem.

Za razliko od prejšnjih metod, ki napovedujejo razliko v rezultatu, sta Baker in McHale [35] modelirala točen rezultat tekem ameriškega nogometa. V ta namen sta uporabila časovno zvezni rojstni proces, pri čemer vsak rojstni dogodek ustreza enemu izmed različnih načinov doseganja točk (ene ali druge ekipe). Intenzivnosti rojstnih dogodkov sta modelirala v odvisnosti od karakteristik ekip (uteženo povprečje opisnih statistik iz prejšnjih tekem), prednosti domačega igrišča in trenutnega stanja (vektor števcev posameznih načinov doseženih točk obeh ekip). Eksperimentalna evalvacija na

treh sezonah NFL (2006-2008) je pokazala, da je model primerljiv s stavnicami.

Goldner [36] je uporabil play-by-play podatke z opisi 1280 tekem iz petih sezon NFL (2005-2009) za modeliranje poteka napada. Uporabil je markovski model s 349 stanji, od tega 9 absorbirajočih stanj za označevanje različnih načinov za dokončanje napada (osvojitev točk, menjava posesti žoge, konec polčasa oziroma tekme). Neabsorbirajoča stanja podajajo kontekst akcije: trenutni poskus, zahtevano razdaljo do novega prvega poskusa in položaj na igrišču (razdalja do končne cone)[2]. Goldner je uporabil model, da vsakemu stanju pridruži pričakovano število doseženih točk, ki predstavlja podlago za analizo učinkovitosti in izbiro strategije napada.

Modeli za napovedovanje izida upoštevajo tudi značilnosti ciljnega športa. Za nogomet je značilno, da so zadetki relativno redki in je povprečno število doseženih golov na tekmi majhno. Iz tega razloga je pri nogometu običajni pristop uporaba Poissonove porazdelitve za modeliranje števila zadetkov, ki se dobro prilega empiričnim podatkom.

Maher [37] je končni rezultat nogometne tekme modeliral z dvema neodvisnima slučajnima spremenljivkama porazdeljenima po Poissonu, pri čemer je parameter posamezne porazdelitve izražen v funkciji napadalne in obrambne moči ekipe. Moči ekip so ocenjene ločeno za domače in gostujoče tekme. Maher je obravnaval tudi uporabo dvorazsežne Poissonove porazdelitve in je eksperimentalno določil vrednost 0.2 kot najbolj ustrezno nastavitev za korelacijski koeficient med domačimi in gostujočimi goli.

Dixon in Coles [38] sta preučevala odvisnost med številom zadetkov domače in gostujoče ekipe. Ugotovila sta, da je odvisnost izražena predvsem v tekmah z malo zadetki, kjer nobena ekipa ni dosegla več kot enega gola. Zato sta predlagala prilagoditev Maherjevega modela, ki omogoča ločeno obravnavo tekem z nizkim rezultatom. V model sta dodala prednost domačega igrišča in opisala način posodabljanja neodvisnih parametrov, ki ponazarja spreminjanje moči ekip skozi sezono. Napovedi modela, naučenega na tekmah angleških ligaških in pokalnih tekmovanj v sezonah 1992-1995, sta testirala na sezoni 1995/1996 in ugotovila, da imajo potencial za pozitiven donos na stavnicah.

Dixon in Robinson [39] sta preučevala čas med zadetki med potekom nogometne tekme. Analizirala sta podatke iz angleških tekmovanj v sezonah 1993-1996 in ugotovila, da frekvenca zadevanja linearno narašča s časom tekme (kar sta pripisala utrujenosti

[2] razdalje so diskretizirane s korakom 5 jardov, zahtevane razdalje do novega prvega poskusa, ki so nad 20 jardov, so združene v eno samo vrednost.

igralcev, ki delajo več napak v obrambi) in je odvisna od trenutnega rezultata. Vodilni ekipi frekvenca zadevanja pada, medtem ko ekipi v zaostanku narašča. Zadetke sta modelirala z nehomogenim dvorazsežnim rojstnim procesom z intenzivnostmi doseganja golov v funkcijski odvisnosti od zmogljivosti ekip, prednosti domačega igrišča, časa in trenutnega rezultata tekme.

Rue in Salvesen [40] sta prav tako izhajala iz predpostavke, da je število zadetkov na nogometni tekmi porazdeljeno po Poissonu, a sta dinamiko spreminjanja moči ekip v času modelirala z bayesovskim pristopom. Z metodo Monte Carlo z markovskimi verigami (ang. Monte Carlo Markov Chain) so na podlagi odigranih tekem posodabljali ocene moči za vse ekipe hkrati.

Titman in sod. [41] so v model soodvisnosti števila zadetkov domače in gostujoče ekipe vključili tudi vpliv pokazanih kartonov na končni izid tekme. Razvoj rezultata so modelirali kot večrazsežni proces štetja, pri čemer so za opredelitev intenzitete posameznih dogodkov uporabili Weibullovo porazdelitev, relativne moči ekip pa so ocenili s kvotami stavnic. Model, naučen na tekmah angleških ligaških tekmovanj v sezonah 2009-2011, je dobro zajel dinamiko vpliva rdečih kartonov na frekvenco zadetkov.

Constantinou in sod. [42] so uporabili Bayesovo mrežo za verjetnostno napovedovanje diskretnega izida nogometne tekme (zmaga domačina, neodločen izid, zmaga gosta). Moči ekip so določili glede na osvojeno število točk v prvenstvu. Verjetnosti izida tekme so modelirali v odvisnosti od objektivnih (določenih na podlagi dejanskih rezultatov iz trenutne in prejšnjih sezon) in subjektivnih faktorjev (ekspertno mnenje). Za učenje modela so uporabili 6244 tekem iz elitnega angleškega ligaškega tekmovanja (sezone od 1993/1994 do 2009/2010), kvaliteto napovedi pa so testirali na 380 tekmah sezone 2010/2011. Uspešnost napovedovanja je bila primerljiva s stavnicami.

Za napovedovanje diskretnega izida se uporabljajo tudi probit regresijski modeli, na primer za nogomet ([43], [44]) in košarko [45].

2.4 *Podrobnejši opis najbolj sorodnih del*

Kljub spodbudnim rezultatom pri modeliranju športnih tekem pa številne raziskave kažejo, da so kvote stavnic še vedno najboljši napovedovalec končnega izida ([46], [47], [48]). Po drugi strani statistični modeli omogočajo generiranje verodostojnih simulacij in pričakovanega razvoja dogodkov na hipotetičnih tekmah med povprečnimi ali določenimi nasprotniki.

V tem razdelku podajamo podrobnejši opis raziskav, ki se ukvarjajo z razvojem dogodkov na športnih tekmah in so neposredno povezana s pričujočim delom. Raziskovalci so največ pozornosti posvetili košarki, ki je zaradi visoke dinamike igre in velikega števila dogodkov na posamezni tekmi zelo primerna za tovrstno modeliranje.

Stern [49] je uporabil 493 tekem ameriške profesionalne košarkarske lige (NBA) ter na podlagi trenutnih izidov ob koncu posameznih četrtin zgradil model Brownovega gibanja razvoja rezultata. Model se je izkazal kot dober napovedovalec verjetnosti zmage na osnovi podane trenutne razlike v rezultatu. Model ni upošteval moči ekip, niti ni modeliral posesti žoge.

Goldman in Rao [50] sta uporabila podoben model za preučevanje efekta psihološkega pritiska na kvaliteto izvedbe. Modelirali so tudi stabilnost napovedane verjetnosti zmage v odvisnosti od majhnih sprememb trenutnega rezultata in jo uporabili za ocenjevanje pomembnosti situacije na končni razplet tekme.

Gabel in Redner [51] sta uporabila model naključnega sprehoda za opisovanje različnih statističnih lastnosti gibanja rezultata med potekom košarkarske tekme. Pokazala sta, da je igralni čas med zadetki eksponentno porazdeljen in je proces doseganja točk brez spomina. Ugotovila sta tudi, da imajo lastne kvalitete ekip majhen vpliv na splošno sliko gibanja rezultata.

Merrit in Clauset [52] sta modelirala dinamiko gibanja rezultata na povprečni tekmi v različnih timskih športih (košarka, ameriški nogomet in hokej). Uporabila sta dva stohastična procesa. Prvi proces proizvaja zadetke, medtem ko drugi proces določa, katera ekipa je dosegla točke. Ugotovila sta tudi, da je model sposoben zelo natančno predvideti končni izid tekme samo na podlagi opažene dinamike nekaj prvih zadetkov.

Shirley [53] je modeliral razvoj košarkarske tekme z diskretno markovsko verigo v prostoru stanj, ki opisujejo, na kakšen način je ekipa pridobila oziroma obdržala žogo. Stanje je zakodirano v obliki trirazsežnega vektorja. Prva komponenta določa ekipo, ki ima žogo (dve možni vrednosti, ki označujeta domačina oziroma gosta). Druga komponenta določa način, na katerega je ekipa prišla do žoge (pet možnih vrednosti, ki označujejo vračanje žoge v igro, ukradeno žogo, skok v napadu, skok v obrambi in pristop k izvedbi prostih metov). Tretja komponenta predstavlja število točk, doseženih ob prihodu v to stanje (štiri možne vrednosti, to so 0, 1, 2 ali 3 točke). Izmed teoretičnih 40 stanj ($2 \times 5 \times 4$), je zaradi logičnih omejitev in pravil igre samo 30 dejansko možnih. Shirley je za vsako ekipo zgradil ločen model na podlagi njenih opisnih statistik in pokazal, da je tak model dober napovedovalec verjetnosti zmage opazovane

ekipe proti povprečnemu nasprotniku. Shirley je opisal tudi možnost učenja modela iz opisov poteka tekem (in ne na podlagi opisnih statistik) ob upoštevanju moči posameznih ekip, toda zaradi pomanjkanja podatkov tega ni izvedel.

Štrumbelj in Vračar [25] sta izhajala iz Shirleyevega modela s to razliko, da sta implicitno razširila prostor stanj z atributi za opis moči ekip. Na ta način sta matriko prehoda med stanji Shirliyevega modela izrazila v odvisnosti od karateristik ekip in tako pridobila splošen model za simuliranje tekem med poljubnima nasprotnikoma. Za opisovanje moči ekip sta uporabila zbirne statistike na podlagi štirih faktorjev [19]. Verjetnosti prehodov med stanji sta modelirala z multinomialno logistično regresijo, pri čemer sta vsako vrstico tranzicijske matrike obravnavala kot ločen klasifikacijski problem (stanja so neodvisna zaradi markovske lastnosti procesa). Na ta način sta dobila 30 klasifikatorjev - po enega za vsako stanje Shirliyevega modela. Izdelava simulatorja tekem med podanima ekipama poteka tako, da se najprej karakteristike ekip uporabijo kot vhod v naučene klasifikatorje, nato pa se vrnjene verjetnostne predikcije združijo (po vrsticah) v homogeno matriko prehoda Shirleyevega modela. Simulacijo tekme med tema ekipama je sedaj možno generirati z naključnim sprehodom po prostoru stanj, v skladu z dobljeno tranzicijsko matriko - zaporedje obiskanih stanj Shirleyevega modela predstavlja razvoj dogodkov na simulirani tekmi. Ker stanja modela ne vsebujejo informacije o poteku igralnega časa, je simulacije konec po vnaprej določenem številu prehodov. Avtorja sta opisano metodo testirala na tekmah rednega dela sezon lige NBA 2007/2008 in 2008/2009 in ugotovila, da je model primeren za modeliranje košarkarskih tekem, njegove napovedi primerljive z drugimi statističnimi pristopi, hkrati pa omogoča globlji vpogled v košarko. Izkazalo se je tudi, da je model pristranski pri ocenjevanju verjetnosti zmage, in sicer precenjuje možnosti slabih in podcenjuje možnosti močnih ekip. Avtorja sta to pripisala homogenosti modela - ker se verjetnostna porazdelitev prehodov med stanji ne spreminja z razvojem tekme, je s tem onemogočeno modeliranje dinamike košarkarske tekme.

Za razliko od zgoraj opisanih metod [53] in [25], ki s prehodi med stanji modelirata spremembo posesti žoge in število doseženih košev med prejšnjo posestjo, v disertaciji uporabimo bolj neposreden pristop in osnovne dogodke modeliranega športa ekplicitno vključimo v opis prostora stanj. Po [25] prevzamemo idejo o parametrizaciji verjetnosti prehodov med stanji z značilkami za opis zmogljivosti ekip. Idejo nadgradimo z dodatno razširitvijo opisa prostora stanj, ki omogoča simuliranje vseh relevantnih vidikov razvoja tekme. V konkretnem primeru modeliranja košarke, kjer metodi [53] in

[25] napovedujeta le zaporedje dogodkov na tekmi, pristop v disertaciji modelira tudi potek igralnega časa med napovedanimi dogodki. Ključna novost pristopa v disertaciji je dodatna razširitev opisa stanja s parametri, ki podajajo kontekst trenutka v razvoju tekme. Za razliko od pristopov [53] in [25], ki gradita homogeno markovsko verigo, pristop v disertaciji omogoča spremenljive verjetnosti prehoda v odvisnosti od časa in trenutne razlike v rezultatu in na ta način omili predpostavko o markovski lastnosti modela. Za razliko od metode [25], ki za opisovanje zmogljivosti ekip uporablja ekspertne atribute, v disertaciji podajamo izvirni postopek za avtomatsko generiranje interpretabilnega atributnega prostora.

3

Modeliranje tekem v športnih igrah

Pri preučevanju dinamičnih sistemov pogosto velja, da nimamo popolne informacije o vseh relevantnih dejavnikih, ki vplivajo na sistem, oziroma ne znamo dovolj natančno opisati, na kakšen način nanj vplivajo. Običajni način reševanja tega problema je, da spreminjanje opazovanega sistema modeliramo kot stohastični proces, pri čemer nepopolno informacijo obravnavamo kot vir naključja v sistemu.

Zaradi enostavnosti modela se bomo zadovoljili s predstavitvijo dinamičnega sistema s pomočjo diskretne markovske verige. Le-ta je stohastični proces, ki se v diskretnih korakih sprehaja med stanji iz podane množice stanj v skladu s podanimi verjetnostmi prehodov. Pri tem velja markovska lastnost procesa, ki določa, da je verjetnost prehoda v naslednje stanje odvisna samo od trenutnega stanja, kar pomeni, da je proces brez spomina. Slednjo omejitev je možno nekoliko omiliti z razširjenim opisom stanja, ki vključuje tudi del opisa zgodovine, relevantne za nadaljevanje procesa.

3.1 Generator tekem povprečnih nasprotnikov

Na razvoj tekme lahko gledamo kot na eno realizacijo stohastičnega procesa. Potek tekme lahko zapišemo kot markovsko verigo $\{\mathbf{X_t}, t \in \mathbb{N}_0\}$ nad nekim prostorom stanj $\mathscr{S}$. V najbolj preprostem primeru predstavlja $\mathscr{S} = \{evt_1, evt_2, \dots, evt_r\}$ končno množico dogodkov, relevantnih za opazovani šport. Tabela 3.1 predstavlja primer množice dogodkov za košarko.

Tabela 3.1

Množica dogodkov za košarko.

Oznaka dogodka	Opis
AJB, HJB	Gostujoča / Domača ekipa je dobila sodniški met.
AINB, HINB	Gostujoča / Domača ekipa je vrnila žogo v igro.
AORB, HORB	Gostujoča / Domača ekipa je dobila skok v napadu.
AOREB, HOREB	Gostujoča / Domača ekipa je dobila ekipni skok v napadu.
ADRB, HDRB	Gostujoča / Domača ekipa je dobila skok v obrambi.
ADREB, HDREB	Gostujoča / Domača ekipa je dobila ekipni skok v obrambi.
A2PA, H2PA	Gostujoča / Domača ekipa je zgrešila met za 2 točki.
A2PM, H2PM	Gostujoča / Domača ekipa je zadela met za 2 točki.
A3PA, H3PA	Gostujoča / Domača ekipa je zgrešila met za 3 točke.
A3PM, H3PM	Gostujoča / Domača ekipa je zadela met za 3 točke.
AFT*i*A, HFT*i*A	Gostujoča / Domača ekipa je zgrešila prvega izmed i preostalih prostih metov ($i \in [1 \dots 3]$).
AFT*i*M, HFT*i*M	Gostujoča / Domača ekipa je zadela prvega izmed i preostalih prostih metov ($i \in [1 \dots 3]$).
ATO, HTO	Gostujoča / Domača ekipa je zapravila žogo.
AV, HV	Gostujoča / Domača ekipa je storila prekršek.
AF, HF	Gostujoča / Domača ekipa je storila osebno napako.

Markovski proces razvoja dogodkov na tekmi lahko opišemo s pomočjo matrike prehoda stanj $P = [p_{ij}]_{i,j=1,2,\dots,r}$, kjer je $p_{ij} = P(\mathbf{X_{t+1}} = evt_j | \mathbf{X_t} = evt_i)$. Posamezne verjetnosti v matriki P lahko ocenimo na podlagi učne množice, ki predstavlja dejanske realizacije stohastičnega procesa (sosledje dogodkov na odigranih tekmah). Konkretno, če z n_{ij} označimo število primerov, ko stanju evt_i sledi stanje evt_j, lahko elemente p_{ij} ocenimo z izrazom:

$$\hat{p}_{ij} = \frac{n_{ij}}{\sum_{k=1}^{r} n_{ik}} \tag{3.1}$$

Večja kot je učna množica, bolj zanesljivo lahko določimo elemente matrike prehoda stanj P. V primeru premajhnega števila podatkov lahko namesto relativne frekvence uporabimo Laplaceov popravek [54] ali m-oceno [55].

Simulacijo tekme (možno realizacijo stohastičnega procesa) dobimo z vzorčenjem markovske verige $(\mathbf{x_0}, \mathbf{x_1}, \dots, \mathbf{x})$ v skladu z matriko prehoda stanj P, kjer $\mathbf{x_0}$ predstavlja začetek tekme in $\mathbf{x}$ njen konec. Pričakovano dolžino tekme (število korakov ω)

določimo na podlagi učne množice.

Dogodki na tekmi niso povsem naključni, temveč so posledica različnih dejavnikov (pravila igre, običajne prakse, značilnosti ekip in podobno). Ob zadosti veliki učni množici bodo ti dejavniki implicitno zajeti v matriki prehoda stanj P, kar pomeni, da bi bilo v principu možno zgraditi generator tekem v poljubno definiranem kontekstu. Na primer, če želimo generirati tekme med točno določenimi nasprotniki, potrebujemo le zadosti veliko učno množico medsebojnih tekem teh dveh ekip. Toda v praksi je to običajno neizvedljivo, saj ekipe odigrajo le nekaj medsebojnih tekem (ali celo nobene) v nekem relevantnem časovnem okviru. Za zanesljivo oceno matrike prehodov stanja P potrebujemo veliko učno množico, sestavljeno iz vseh odigranih tekem, ne glede na nasprotnike. Posledično se izgubi specifičnost posameznih ekip in dobimo generator tekem povprečnih nasprotnikov.

3.2 Generator tekem med specifičnima nasprotnikoma

Če želimo sestaviti generator tekem med specifičnima ekipama, je matriko prehoda stanj P potrebno izraziti v odvisnosti od izbranega nabora značilk, ki opisujejo zmogljivosti teh ekip. V tabeli 3.2 je predstavljen primer značilk za opisovanje zmogljivosti košarkarske ekipe, ki so se v prejšnjih raziskavah izkazale kot informativne ([25], [26]).

Naj vektorja $\mathbf{a}$ in $\mathbf{h}$ označujeta vrednosti izbranih značilk za gostujočo in domačo ekipo. Kodiranje prostora stanj se sedaj razširi v vektor oblike:

$$\langle Evt, \mathbf{a}, \mathbf{h} \rangle$$

kjer z Evt označimo trenutni dogodek in velja $Evt \in \{evt_1, evt_2, \dots, evt_r\}$.

Z opisano razširitvijo prostor stanj postane zvezen. Zaradi enostavnosti predpostavimo, da se zmogljivosti ekip $\mathbf{a}$ in $\mathbf{h}$ ne spreminjajo med trajanjem posamezne tekme (čeprav se lahko spreminjajo tekom sezone). Naj bo $f(\mathbf{y}|\mathbf{x})$ porazdelitev vrednosti slučajne spremenljivke $\mathbf{X_{n+1}}$ v odvisnosti od vrednosti slučajne spremenljivke $\mathbf{X_n}$. Ker je Evt edina spremenljiva komponenta v zapisu stanj $\mathbf{x_i}$, lahko zgornjo pogojno porazdelitev izrazimo v obliki:

$$f(\mathbf{y}|\mathbf{x}) = f_{Evt}(y^{(Evt)}|\mathbf{x}) \tag{3.2}$$

Pogojno porazdelitev iz enačbe 3.2 lahko ocenimo z modelom M_{Evt}, ki poda verjetnostne napovedi naslednjega dogodka ob podanem opisu trenutnega stanja. Simulacijo

Tabela 3.2

Statistike za opis zmogljivosti košarkarske ekipe. Izražene so na podlagi standardnih statističnih podatkov: FG - zadeti meti iz igre (ang. field goals made), FGA - poskusi metov iz igre (ang. field goals attempted), $3P$ - zadeti meti za tri točke (ang. three point shots made), FT - zadeti prosti meti (ang. free throws made), FTA - poskusi prostih metov (ang. free throws attempted), TOV - izgubljene žoge (ang. turnovers), ORB - skoki v napadu (ang. offensive rebounds) in DRB - skoki v obrambi (ang. defensive rebounds). Izkazujejo uspešnost meta iz igre ($EFG\%$), odstotek zadetih prostih metov ($FTF\%$), delež izgubljenih žog na posest ($TOVr$), delež dobljenih skokov v napadu ($ORBr$) in obrambi ($DRBr$). Prefiks o označuje, da se statistika nanaša na učinkovitost nasprotnega moštva.

Effective Field Goal %	*Turnover Ratio*
$EFG\% = \frac{FG + \frac{1}{2}3P}{FGA}$	$TOVr = \frac{TOV}{FGA + TOV + 0.44 \cdot FTA}$
Off. Rebound Ratio	*Def. Rebound Ratio*
$ORBr = \frac{ORB}{ORB + oDRB}$	$DRBr = \frac{DRB}{DRB + oORB}$
Free Throw Factor	*opponents EFG%*
$FTF\% = \frac{FT}{FTA}$	$oEFG\% = \frac{oFG + \frac{1}{2}o3P}{oFGA}$
opponents TOV	*opponents FTF*
$oTOVr = \frac{oTOV}{oFGA + oTOV + 0.44 \cdot oFTA}$	$oFTF\% = \frac{oFT}{oFTA}$

poteka tekme generiramo tako, da vzorčimo zaporedje dogodkov iz predikcij modela M_{Evt}.

3.3 Generator tekem z upoštevanjem konteksta

Osnovna pomanjkljivost prej opisanega modela za generiranje tekem med specifičnima nasprotnikoma je njegova statičnost. Zaradi homogenosti matrike prehoda stanja P (verjetnosti p_{ij} se ne spreminjajo skozi čas), je dinamika generiranih dogodkov enaka tekom celotne tekme. Empirične študije ([25], [51], [52]) podpirajo ekspertno znanje in intuicijo, da se dinamika igre spreminja v odvisnosti od trenutnega konteksta, ki ga v prvi vrsti določata čas do konca tekme in trenutni izid (na primer, ekipa z minimalnim rezultatskim zaostankom bo v zadnjih minutah igrala bistveno drugače kot na polovici tekme). Pri nekaterih športnih igrah ima pomembno vlogo tudi položaj na igrišču (na primer, v nogometu je bistvena razlika, ali je ekipa dobila prosti strel 20m ali 60m od nasprotnikovih vrat). Lokalni kontekst tekme definirajo tudi različni števci, ki izhajajo iz pravil igre (na primer, v ameriškem nogometu ima ekipa na voljo štiri poskuse za osvojitev vsaj 10 jardov; pristop pri prvem poskusu se bistveno razlikuje od tistega pri četrtem poskusu).

Neupoštevanje odvisnosti matrike P od lokalnega konteksta povzroča generiranje nerealističnih simulacij [26]. Pomanjkljivost odpravimo na podoben način kot pri prejšnjem modelu. Naj bodo $A_1, A_2, \ldots, A_m$ slučajne spremenljivke, ki neposredno vplivajo na izvajanje stohastičnega procesa (razvoj tekme). Naj bodo $B_1, B_2, \ldots, B_n$ spremenljivke, ki vplivajo na izvajanje stohastičnega procesa, vendar niso slučajne (na primer, trenutni rezultat tekme ima vpliv na nadaljnji potek dogodkov, je pa posledica prejšnjih dogodkov in ni naključna spremenljivka). Naj bodo $C_1, C_2, \ldots, C_k$ spremenljivke, ki vplivajo na izvajanje procesa, vendar se njihova vrednost ne spreminja (v to kategorijo lahko uvrstimo značilke, ki opisujejo zmogljivosti ekip). Kodiranje prostora stanj lahko zapišemo v obliki vektorja:

$$\langle A_1, A_2, \ldots, A_m, B_1, B_2, \ldots, B_n, C_1, C_2, \ldots, C_k \rangle$$

Z uporabo verižnega pravila lahko pogojno porazdelitev $f(\mathbf{y}|\mathbf{x})$ izrazimo v obliki:

$$\begin{aligned} f(\mathbf{y}|\mathbf{x}) = & f_{A_1}(y^{(A_1)}|\mathbf{x}) \cdot f_{A_2}(y^{(A_2)}|\mathbf{x}, y^{(A_1)}) \cdots \\ & \cdots f_{A_m}(y^{(A_m)}|\mathbf{x}, y^{(A_1, A_2, \ldots, A_{m-1})}) \end{aligned} \tag{3.3}$$

Modeliranje pogojne porazdelitve iz enačbe 3.3 izvedemo s kaskado zaporedno vezanih modelov $M_1, M_2, \ldots, M_m$. Prvi v kaskadi, model M_1, oceni marginalno pogojno porazdelitev slučajne spremenljivke A_1 na podlagi podanega opisa trenutnega stanja $\mathbf{x}$. Vsak naslednji model M_i ocenjuje porazdelitev slučajne spremenljivke A_i na podlagi trenutnega stanja $\mathbf{x}$ in pogojeno s komponentami $A_1, \ldots A_{i-1}$ iz opisa naslednjega stanja $\mathbf{y}$. Vzorčenje iz porazdelitve $f(\mathbf{y}|\mathbf{x})$ izvedemo z vzorčenjem modeliranih marginalnih porazdelitev, pri čemer vsak model kot vhod prejme opis trenutnega stanja in izbrane vzorce vseh predhodnih modelov v kaskadi. Ko izberemo vzorce iz celotne kaskade, so postavljene vrednosti komponent $A_1, \ldots A_m$ opisa naslednjega stanja $\mathbf{y}$. Sedaj postavimo tudi vrednosti komponent $B_1, \ldots B_n$, saj le-te lahko rekonstruiramo neposredno iz komponent $A_1, \ldots A_m$. Ker so vrednosti komponent $C_1, \ldots, C_k$ ves čas konstantne, je s tem določen celoten opis naslednjega stanja $\mathbf{y}$.

Modelov iz kaskade se lahko naučimo vsakega posebej, saj so medseboj neodvisni. Načeloma bi bilo možno uporabiti katerikoli parametrični ali neparametrični model za oceno pogojnih porazdelitev iz enačbe 3.3, a potrebno je upoštevati, da razvoj tekme poteka v skladu z diskretnimi pravili športa, ki jih generirane simulacije ne smejo kršiti. Pravila športa določajo podmnožice prostora stanj, ki so možni nasledniki trenutnega stanja. To vodi k naravnemu razbitju celotnega prostora stanj na disjunktne podmnožice. Torej, če se želimo izogniti neveljavnim prehodom med stanji, je potrebno hierarhično razdeliti prostor stanj in potem modelirati vsak podprostor posebej. To hierarhično delitev lahko izvedemo avtomatsko ali ročno, z uvedbo domenskega predznanja. Na podlagi teh ugotovitev sledi, da so drevesa (odločitvena in regresijska) primeren model za modeliranje razvoja športnih dogodkov. Dodatna prednost drevesnih modelov je njihova razumljivost in verifikacija s strani domenskega eksperta.

Ko so modeli iz kaskade naučeni, lahko generiramo simulacije tekem med specifičnima ekipama iz poljubno definirane izhodiščne situacije (glej algoritem 1). Funkcija *gameEnded* v vrstici 5 sprejme opis trenutnega stanja in preveri, ali so izpolnjeni pogoji za zaključek tekme (na primer, ali je potekel igralni čas). V vrstici 7 se postavljajo vrednosti slučajnih spremenljivk A_i z vzorčenjem iz pogojnih porazdelitev, kot jih vrnejo modeli M_i. Funkcija *update* v vrstici 9 na podlagi izbranih vzorcev rekonstruira komponente $B_1, \ldots, B_n$ v opisu novega stanja. Dobljeni opis novega stanja se doda na konec seznama generiranih dogodkov v vrstici 10.

Algoritem 1

Simulacija tekme med specifičnima ekipama

Vhod:

izhodiščno stanje $\mathbf{s}$,

kaskada modelov M_i za oceno $f_{\mathbf{Y}^{(A_i)}|\mathbf{X},\mathbf{Y}^{(A_1,\ldots,A_{i-1})}}$.

Izhod:

generirano zaporedje stanj *outSeq*.

1: *function* SIMULATE($\mathbf{s}, M_1, \ldots, M_m$)
2: $outSeq \leftarrow \emptyset$
3: $\mathbf{x} \leftarrow \mathbf{s}$
4: $\mathbf{y} \leftarrow \mathbf{x}$
5: *while not gameEnded*($\mathbf{x}$) *do*
6: *for* $i = 1$ to m *do*
7: $\mathbf{y}^{(A_i)} \leftarrow$ sample from $M_i(\mathbf{x}, \mathbf{y}^{(A_1,\ldots,A_{i-1})})$
8: *end for*
9: $\mathbf{y}^{(B_1,\ldots,B_n)} \leftarrow update(\mathbf{y}^{(A_1,\ldots,A_m)}, \mathbf{x})$
10: $outSeq \leftarrow append(outSeq, \mathbf{y})$
11: $\mathbf{x} \leftarrow \mathbf{y}$
12: *end while*
13: *return outSeq*
14: *end function*

4

Avtomatsko generiranje atributov

V poglavju 3 smo predstavili osnovno metodologijo za modeliranje razvoja športnih tekem. Od modela za generiranje realističnih simulacij tekem med dvema specifičnima ekipama pričakujemo, da se bodo v generiranem izhodu (zaporedju dogodkov) zrcalile zmogljivosti teh ekip. To dosežemo tako, da modeliramo odvisnost vrednosti matrike prehoda stanj od trenutnega konteksta tekme, ki vključuje tudi opise zmogljivosti ekip.

Pri izbiri atributnega prostora za opis zmogljivosti ekip lahko uporabimo ekspertno znanje, saj so za večino najpopularnejših športov definirani različni kazalci uspešnosti. Težava je v tem, da ti kazalci uspešnosti praviloma temeljijo na človeškem dojemanju koncepta zmogljivosti ekip in so bolj namenjeni predikciji končnega zmagovalca, kar ni nujno povezano z napovedovanjem, kaj (ter kje in kdaj) se bo na tekmi naslednje zgodilo.

V nadaljevanju predstavljamo postopek za avtomatsko konstrukcijo atributnega prostora za opis zmogljivosti ekip, ki je namenjen modeliranju razvoja dogodkov na tekmi in hkrati omogoča interpretabilnost zgrajenih atributov. Slednja lastnost zahteva eksplicitno podano in dokaj enostavno strukturo konstruiranih atributov, kar prinaša vrsto prednosti. Na prvem mestu je tu praktičnost uporabe, saj razumljivi atributi omogočajo enostaven opis zmogljivosti ekip (tudi hipotetičnih nasprotnikov) brez potrebe po natančni oceni iz velike in skrbno pripravljene učne množice. Eksplicitna definicija atributov omogoča tudi analizo, kako določene spremembe v igri neke ekipe vplivajo na pričakovan potek tekme ter s tem tudi na možnost za končno zmago. Nenazadnje, interpretabilen atributni prostor omogoča nov vpogled in boljše razumevanje obstoječih (ali celo novo odkritih) konceptov športne igre, ki jo modeliramo.

4.1 Osnovna ideja

Podana je množica dogodkov $\mathscr{S} = \{evt_1, evt_2, \ldots, evt_r\}$, ki jim sledimo na tekmah ciljnega športa. V razdelku 3.1 smo predstavili pristop k modeliranju razvoja dogodkov na tekmi v obliki markovskega procesa, ki je definiran z matriko prehoda stanj $P = [p_{ij}]_{i,j=1,2,\ldots,r}$, kjer je $p_{ij} = P(\mathbf{X_{t+1}} = evt_j | \mathbf{X_t} = evt_i)$. Posamezne elemente matrike prehoda p_{ij} lahko ocenimo z uporabo enačbe 3.1 na podlagi učne množice, sestavljene iz sekvenc dogodkov, ki predstavljajo dejanske realizacije stohastičnega procesa.

V primerih, ko je število različnih dogodkov r majhno (tipično ≤ 4), so elementi matrike prehoda p_{ij} primerna izbira za atributni prostor, saj natančno opisujejo razvoj stohastičnega procesa (tekme) in imajo jasen pomen (torej so interpretabilni). Toda v resničnih športih je število relevantnih dogodkov veliko (tipično > 10), zaradi česar

postane r^2-dimenzionalni atributni prostor nepraktičen za strojno učenje. Posamezni atributi premalo generalizirajo zakonitosti, ki krmilijo stohastični proces, uporabljeni skupaj pa lahko povzročijo preveliko prileganje podatkom.

Osnovna ideja našega pristopa temelji na ocenjevanju verjetnosti prehoda, toda ne med posameznimi stanji, temveč med manjšim številom abstraktnih stanj, ki predstavljajo višjenivojske koncepte - situacije, ki jih opišemo z množico osnovnih stanj. Pričakujemo, da atributi za opisovanje teh višjenivojskih konceptov bolje generalizirajo vzvode, ki vplivajo na potek stohastičnega procesa. Višjenivojske koncepte - abstraktna stanja - bomo identificirali na podlagi podobnosti med osnovnimi stanji, zato potrebujemo kriterij za primerjavo.

Za vsako osnovno stanje $evt_i \in \mathscr{S}$ konstruiramo diskretni verjetnostni porazdelitvi $P_{evt_i} = \{p_j\}_{j=1\ldots r}$ in $Q_{evt_i} = \{q_j\}_{j=1\ldots r}$ z elementi

$$p_j = \frac{n_{ij}}{\sum_{k=1}^{r} n_{ik}}, \qquad q_j = \frac{n_{ij}}{\sum_{k=1}^{r} n_{kj}} \tag{4.1}$$

kjer n_{ij} označuje število primerov iz učnih sekvenc, ko stanju evt_i sledi stanje evt_j. P_{evt_i} opisuje verjetnostno porazdelitev za stanje, ki bo sledilo stanju evt_i, medtem ko Q_{evt_i} opisuje verjetnostno porazdelitev za stanje, ki je predhodnik stanja evt_i. Opisani verjetnostni porazdelitvi lahko uporabimo kot osnovo za določanje podobnosti med osnovnimi stanji. V tabelah 4.1 in 4.2 so prikazani konkretni primeri omenjenih verjetnostnih porazdelitev za nekaj osnovnih stanj v košarki.

Za kvantifikacijo podobnosti med diskretnima porazdelitvama $A = (a_1, \ldots, a_k)$ in $B = (b_1, \ldots, b_k)$ uporabimo Hellingerjevo razdaljo:

$$H(A,B) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{a_i} - \sqrt{b_i})^2} \tag{4.2}$$

Če nas zanima podobnost med stanjema evt_i in evt_j glede na možnega naslednika v sekvenci dogodkov, potem primerjamo verjetnostni porazdelitvi P_{evt_i} in P_{evt_j}. Za oceno podobnosti med tema stanjema glede na prednika v sekvenci pa primerjamo porazdelitvi Q_{evt_i} in Q_{evt_j}. Hellingerjevo razdaljo smo izbrali, ker je metrika, kar pomeni, da je simetrična. Uporaba simetrične razdalje zagotavlja, da sta opazovani stanji evt_i in evt_j enako podobni ne glede na vrstni red primerjave, kar poenostavi postopke, opisane v nadaljevanju besedila.

Tabela 4.1

Primer verjetnostnih porazdelitev P_{evt_i} za osnovna stanja H2PA, H3PA, H2PM, H3PM, A2PA, A3PA, A2PM in A3PM (košarka). Tabela vsebuje verjetnosti prehodov iz omenjenih stanj, pri čemer so zaradi omejenega prostora prikazani samo pari stanj z verjetnostmi prehoda ≥ 0.01. Prikazane vrednosti, dobljene iz tekem rednega dela sezon 2008/2009, 2009/2010 in 2010/2011 lige NBA, so zaokrožene na dve decimalki. Porazdelitve P_{evt_i} predstavljajo vrstice v tabeli. Za primer si oglejmo podobnosti med vrsticami, ki se nanašajo na domačo ekipo in predstavljajo zgrešene (stanji H2PA in H3PA) oziroma zadete mete iz igre (stanji H2PM in H3PM). Po zgrešenih metih so najverjetnejši skoki (obrambna ADRB in ADREB, ter napadna HORB in HOREB), medtem ko po zadetem košu najverjetneje sledi vračanje žoge v igro s strani gostujoče ekipe (AINB). Podoben premislek velja tudi za skupino stanj, ki predstavljajo zgrešene in zadete mete iz igre s strani gostujoče ekipe (A2PA, A3PA, A2PM in A3PM).

	ADRB	ADREB	AF	AINB	AORB	AOREB	HDRB	HDREB	HF	HINB	HORB	HOREB
H2PA	0.63	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.26	0.06
H3PA	0.64	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.22	0.09
H2PM	0.00	0.00	0.08	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
H3PM	0.00	0.00	0.01	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A2PA	0.00	0.00	0.00	0.00	0.25	0.06	0.64	0.04	0.00	0.00	0.00	0.00
A3PA	0.00	0.00	0.00	0.00	0.21	0.09	0.64	0.06	0.00	0.00	0.00	0.00
A2PM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.92	0.00	0.00
A3PM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.99	0.00	0.00

Tabela 4.2

Primer verjetnostnih porazdelitev Q_{evt_i} za osnovna stanja H2PA, H3PA, H2PM, H3PM, A2PA, A3PA, A2PM in A3PM (košarka). Tabela vsebuje verjetnosti prehodov v omenjena stanja, pri čemer so zaradi omejenega prostora prikazani samo pari stanj z verjetnostmi prehodov ≥ 0.01. Prikazane vrednosti, dobljene iz tekem rednega dela sezon 2008/2009, 2009/2010 in 2010/2011 lige NBA, so zaokrožene na dve decimalki. Porazdelitve Q_{evt_i} predstavljajo stolpce v tabeli. Za primer si oglejmo podobnosti med stolpci, ki ustrezajo stanjem H2PA, H3PA, H2PM in H3PM (zgrešeni in zadeti meti iz igre s strani domače ekipe). Prihod v ta stanja je možen samo v primeru, ko ima domača ekipa žogo. Domačin do žoge pride z uspešnim skokom (HDREB, HDREB, HORB, HOREB), z vračanjem žoge v igro (HINB), oziroma tako, da gostujoča ekipa zapravi žogo (ATO). Podoben premislek velja tudi za skupino stanj, ki predstavljajo zgrešene in zadete mete iz igre s strani gostujoče ekipe (A2PA, A3PA, A2PM in A3PM).

	H2PA	H3PA	H2PM	H3PM	A2PA	A3PA	A2PM	A3PM
ADRB	0.00	0.00	0.00	0.00	0.25	0.28	0.26	0.29
ADREB	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01
AINB	0.00	0.00	0.00	0.00	0.56	0.57	0.50	0.56
AORB	0.00	0.00	0.00	0.00	0.10	0.07	0.13	0.07
AOREB	0.00	0.00	0.00	0.00	0.02	0.01	0.01	0.01
ATO	0.04	0.05	0.08	0.05	0.00	0.00	0.00	0.00
HDRB	0.25	0.29	0.26	0.30	0.00	0.00	0.00	0.00
HDREB	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00
HINB	0.56	0.56	0.49	0.54	0.00	0.00	0.00	0.00
HORB	0.11	0.07	0.13	0.07	0.00	0.00	0.00	0.00
HOREB	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00
HTO	0.00	0.00	0.00	0.00	0.05	0.05	0.08	0.05

Vsaka množica $Y \subseteq \mathscr{S}$, ki je sestavljena iz osnovnih stanj s podobnimi verjetnostnimi porazdelitvami Q_{evt_i} (tj. osnovnih stanj s podobnimi predniki), predstavlja nek višjenivojski koncept. Ta koncept lahko interpretiramo kot pogoj, da razvoj stohastičnega procesa pride v enega izmed stanj iz množice Y. Po drugi strani tudi vsaka množica $X \subseteq \mathscr{S}$, sestavljena iz osnovnih stanj s podobnimi verjetnostnimi porazdelitvami P_{evt_i}, predstavlja nek višjenivojski koncept, ki ga lahko interpretiramo kot posledico ob nastopu nekega stanja iz množice X. Za našo obravnavo so zanimive množice, za katere velja $X \subset Y$, saj v tem primeru obstaja neposredna vzročno-posledična zveza med konceptoma Y in X. Če identificiramo particijo $Y = \{X_1, X_2, \ldots, X_n\}$ tako, da velja $n \geq 2$, $\bigcup_{i=1}^{n} X_i = Y$ in $X_i \bigcap X_j = \varnothing$ za $i \neq j$, dobimo popoln sistem dogodkov, pri čemer vsaka množica X_i predstavlja konceptualno podobne razplete situacije Y.

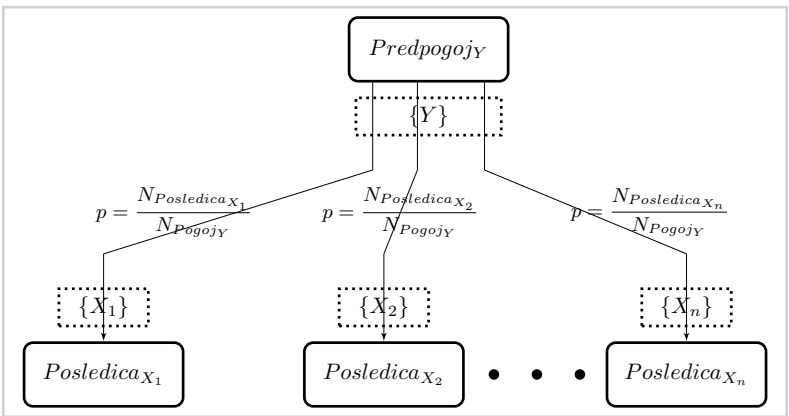
Sedaj je možno razvoj stohastičnega procesa opisati kot sprehajanje med abstraktnimi koncepti: nastop nekega osnovnega dogodka *evt* sproži prehod iz koncepta $Pogoj_Y$ (definiran z množico Y) v koncept $Posledica_X$ (definiran z množico X). To nam omogoča, da za vsak par abstraktnih stanj, ki ustrezajo nekemu pogoju in enemu izmed njegovih posledic, konstruiramo atribut oblike:

$$novAtribut = \frac{N_{Posledica_X}}{N_{Pogoj_Y}} = \frac{\sum\limits_{i:evt_i \in X} \sum\limits_{k=1}^{r} n_{ik}}{\sum\limits_{j:evt_j \in Y} \sum\limits_{k=1}^{r} n_{kj}} \tag{4.3}$$

kjer N_{Pogoj} in $N_{Posledica}$ označujeta število prihodov v ustrezno abstraktno stanje. Prihod v abstraktno stanje identificiramo z nastankom osnovnega dogodka, ki je element množice, s katero je abstraktno stanje definirano. Vrednost tako konstruiranega atributa predstavlja verjetnost nastanka razpleta $Posledica_X$ ob prihodu v abstraktno stanje $Pogoj_Y$. Osnovna ideja konstrukcije atributnega prostora je prikazana na sliki 4.1.

4.2 Identifikacija abstraktnih konceptov

Postopek za identifikacijo abstraktnih stanj temelji na izbranem kriteriju za oceno podobnosti med osnovnimi stanji evt_i. V prejšnjem razdelku smo predstavili dva kriterija: na podlagi podobnosti med diskretnimi porazdelitvami za opis naslednika (poimenovali smo jih P_{evt_i}) oziroma predhodnika stanja (poimenovali smo jih Q_{evt_i}) v razvoju stohastičnega procesa.



Slika 4.1

$Pogoj_Y$ predstavlja abstraktno stanje, ki je definirano z množico Y. Elementi množice Y so osnovna stanja s podobno porazdelitvijo prednika v opazovanih realizacijah procesa. $Pogoj_Y$ lahko interpretiramo kot abstraktni koncept, ki predstavlja pogoj za prihod v neko izmed stanj iz Y. Abstraktna stanja $Posledica_{X_i}$ predstavljajo možna nadaljevanja razvoja procesa po prihodu v stanje $Pogoj_Y$. Vse množice X_i so sestavljene iz osnovnih stanj s podobnimi porazdelitvami naslednika v opazovanih realizacijah procesa. Velja $\bigcup_{i=1}^{n} X_i = Y$ in $X_i \bigcap X_j = \emptyset$ za $i \neq j$. Oznake na povezavah predstavljajo verjetnosti razvoja procesa v ustrezni smeri, pri čemer N_{Stanje} označuje število prihodov opazovanih sekvenc razvoja procesa v neko izmed stanj iz množice, ki opredeljuje to abstraktno stanje.

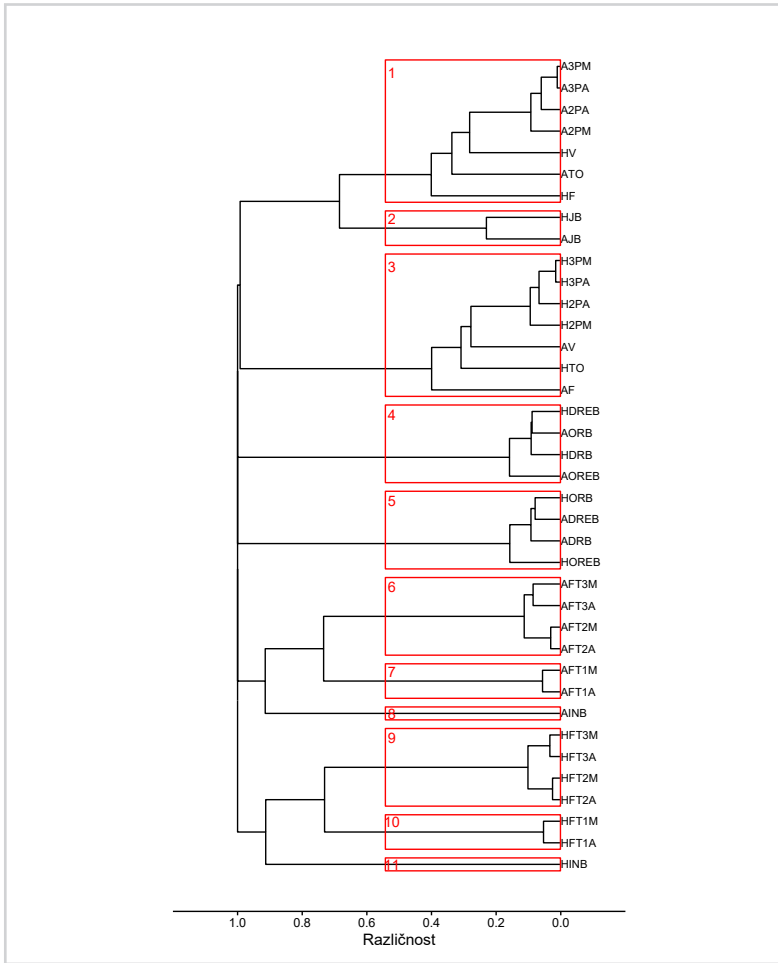
Najprej izračunamo matriko $D = [d_{ij}]_{i,j=1\ldots r}$, kjer element d_{ij} predstavlja Hellingerjevo razdaljo med stanjema i in j glede na izbrani kriterij. Na podlagi matrike razdalj D z uporabo algoritma za hierarhično razvrščanje [56] lahko sestavimo drevo združevanja (dendrogram), ki ponazarja hierarhično ureditev osnovnih stanj v skupine glede na kriterij podobnosti. Listi drevesa združevanja predstavljajo osnovna stanja evt_i, notranja vozlišča pa skupine osnovnih stanj. Skupine v notranjih vozliščih nastanejo z združevanjem skupin, ki ustrezajo poddrevesom tega vozlišča. Višina notranjih vozlišč dendrograma je sorazmerna meri različnosti (razdalji) med skupinami, iz katerih je sestavljena. V našem primeru smo zainteresirani za določitev izoliranih, navznoter kohezivnih skupin osnovnih stanj, zato je najbolj primerna uporaba metode polne povezanosti (ang. complete linkage), ki razdaljo med skupinami določa glede na najbolj različne elemente, ki jih sestavljajo.

Z rezanjem drevesa združevanja na določeni višini (meri različnosti) dobimo razvrstitev osnovnih stanj v skupine, ki so predstavljene z nastalimi poddrevesi. Za tako dobljene skupine velja, da se osnovna stanja v njih ne razlikujejo za več, kot je določeno z višino rezanja. V splošnem je težko določiti najbolj ustrezno višino rezanja, saj število končnih skupin ni vnaprej znano. Okvirno pa velja, da želimo dobiti čim manjše število skupin, ki so sestavljene iz čim bolj podobnih osnovnih stanj. Kot možni hevristiki za izbiro višine rezanja lahko uporabimo metodo po kriteriju minimalne dolžine opisa [57] ali pa silhuetno metodo [58]. Pri metodi za izračun silhuetnih koeficientov, ki ocenjujejo kvaliteto razvrščanja na podlagi podobnosti primerov znotraj skupin in oddaljenosti primerov iz različnih skupin, uporabimo maksimalno razdaljo med primeri (in ne povprečno razdaljo med primeri, kot je to običajno). Na ta način izvedemo bolj konzervativno rezanje dendrograma, ki poda manjše in bolj kohezivne skupine.

Kot primer si poglejmo identifikacijo abstraktnih stanj za košarko na podlagi podobnosti verjetnostne porazdelitve prednikov. Izhajamo iz množice osnovnih stanj $\{evt_i\}$, ki je predstavljena v tabeli 3.1. Pri konstrukciji verjetnostne porazdelitve Q_{evt_i} smo v skladu z enačbo 4.1 uporabili podatke iz 3690 tekem, odigranih v rednem delu sezon 2008/2009, 2009/2010 in 2010/2011 ameriške profesionalne košarkarske lige (NBA). Slika 4.2 prikazuje identificirane abstraktne koncepte, ki smo jih dobili z rezanjem dendrograma z uporabo silhuetne metode. Vsak rdeči okvir označuje množico osnovnih stanj, ki določajo en abstraktni koncept. V prejšnjem razdelku smo ugotovili, da tak abstraktni koncept lahko interpretiramo kot predpogoj za nastanek enega izmed osnovnih dogodkov iz skupine, ki ga sestavlja. Z uporabo domenskega predznanja

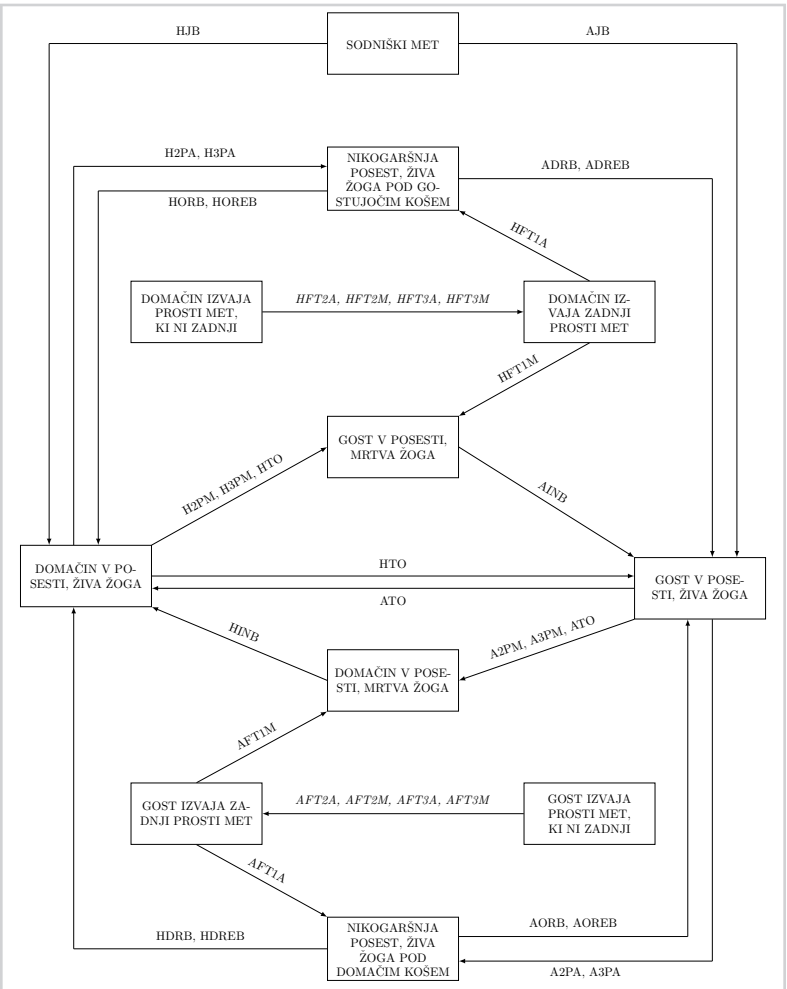
lahko ugotovimo pomen dejanskih konceptov, ki se skrivajo za dobljenimi skupinami. Skupini z oznako 1 in 3 predstavljata koncept posesti žive žoge. Ta koncept je predpogoj za nastop enega izmed naslednjih dogodkov: zgrešen met za dve (stanje A2PA oziroma H2PA) ali tri točke (stanje A3PA oziroma H3PA), uspešen met za dve (stanje A2PM oziroma H2PM) ali tri točke (stanje A3PM oziroma H3PM), izgubljena žoga v napadu (stanje ATO oziroma HTO), prekršek ali osebna napaka v obrambi (stanji HV in HF oziroma AV in AF). Skupini 8 in 11 pa po drugi strani predstavljata koncept posesti žoge, ki ni živa (pri čemer ne gre za izvajanje prostih metov). Ta koncept je predpogoj za začetek nove akcije z vračanjem žoge v igro (stanje AINB oziroma HINB). Skupini 4 in 5 predstavljata koncept žive žoge, ki ni v posesti nobene ekipe. Ta situacija je predpogoj za uspešen skok ekipe v obrambi (stanji ADRB in ADREB oziroma HDRB in HDREB) ali ekipe v napadu (stanji HORB in HOREB oziroma AORB in AOREB). Skupina 2 predstavlja koncept žive žoge po sodniškem metu. Žogo lahko pridobi domača (stanje HJB) ali gostujoča ekipa (stanje AJB). Skupini 7 in 10 predstavljata koncept zadnjega prostega meta v seriji. Situacija se razplete bodisi s košem (stanje AFT1M oziroma HFT1M) bodisi z zgrešenim metom (stanje AFT1A oziroma HFT1A). Po drugi strani, skupini 6 in 9 predstavljata koncept prostega meta, ki ni zadnji v seriji. Glavna razlika glede na skupini 7 in 10 je v tem, da po opravljenem metu žoga ni živa, temveč sledi naslednji prosti met v seriji. Glede na konkretno situacijo sta možna dva razpleta: zadetek (stanji AFT2M in AFT3M oziroma HFT2M in HFT3M) ali pa zgrešeni met (stanji AFT2A in AFT3A oziroma HFT2A in HFT3A). Z domenskim predznanjem lahko sestavimo diagram stanj (glej sliko 4.3), v katerem vozlišča predstavljajo identificirane koncepte. Unija izhodnih povezav iz posameznega vozlišča je enaka naboru osnovnih dogodkov iz ustreznega koncepta (razen povezav HF, HV, AF in AV, ki so odstranjene zaradi večje preglednosti diagrama).

Identificirani abstraktni koncepti predstavljajo le približek dejanskih konceptov. Po rezanju dendrograma se vsako osnovno stanje (dogodek) znajde v natanko eni skupini (torej pripada samo enemu abstraktnemu konceptu). V realnosti pa določeni dogodki lahko nastopajo v različnih konceptih (na primer, do prekrška ali osebne napake lahko pride praktično v vseh fazah igre). Algoritem za hierarhično razvrščanje bo vsak dogodek uvrstil v tisto skupino, za katero je ta dogodek najbolj značilen. Na ta način se izognemo prevelikem številu identificiranih konceptov.



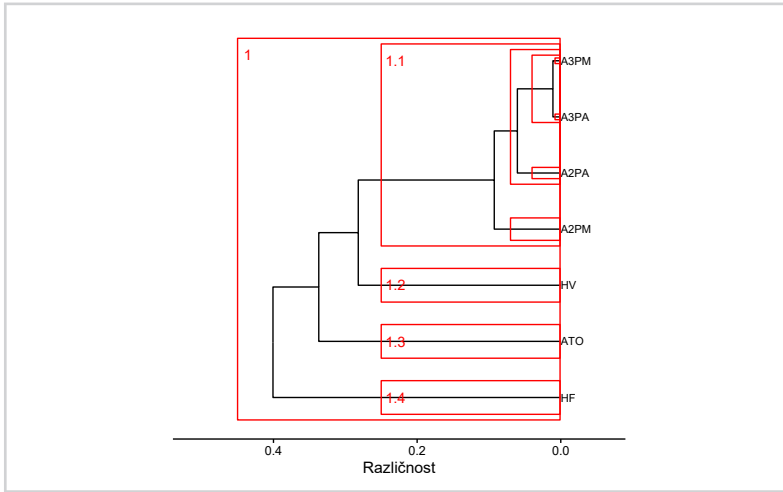
Slika 4.2

Drevo združevanja osnovnih dogodkov za košarko na podlagi podobnosti verjetnostne porazdelitve prednikov. V postopku hierarhičnega združevanja je uporabljena metoda popolne povezanosti. Rdeči okvirji z oznakami 1-11 predstavljajo skupine, dobljene z enkratnim rezanjem dendrograma na podlagi silhuetne metode.



Slika 4.3

Poenostavljeni diagram stanj za razvoj košarkarske tekme. Stanja predstavljajo identificirane koncepte, prikazane na sliki 4.2. Unija izstopnih povezav iz stanj ustreza naboru dogodkov, ki identificirajo posamezne koncepte. Zaradi večje preglednosti diagram ne vsebuje prehodov ob osebnih napakah (dogodki HF in AF) ter prekrškov (HV in AV).



Slika 4.4

Drevo združevanja podmnožice osnovnih dogodkov {A2PA, A2PM, A3PA, A3PM, ATO, HV, HF} na podlagi podobnosti verjetnostne porazdelitve prednikov. Izbrana podmnožica dogodkov ustreza konceptu posesti žive žoge s strani gostujoče ekipe. V postopku hierarhičnega združevanja je uporabljena metoda popolne povezanosti. Rdeči okvirji predstavljajo skupine (podkoncepte), dobljene z rekurzivnim rezanjem dendrograma na podlagi silhuetne metode.

4.2.1 Identifikacija podkonceptov

Rezultat, prikazan na sliki 4.2, predstavlja združevanje osnovnih stanj po podobnosti glede na celotni prostor $\{evt_i\}$. Pri identifikaciji podobnih skupin se lahko omejimo samo na neko podmnožico osnovnih stanj. Z omejevanjem na podmnožice, ki določajo že identificirane koncepte, odkrivamo podkoncepte znotraj bolj splošnih konceptov. Slika 4.4 prikazuje rezultat združevanja stanj pri omejitvi na dogodke, ki opisujejo koncept posesti žive žoge s strani gostujoče ekipe (to je skupina dogodkov iz rdečega okvirja z oznako 1 s slike 4.2). Z uporabo domenskega predznanja lahko tudi tokrat ugotovimo pomen dejanskih konceptov, ki so v ozadju dobljenih skupin. Rdeči okvir z oznako 1.1 ustreza konceptu meta iz igre. Okvir z oznako 1.3 predstavlja koncept izgubljene žoge v napadu. Okvirja z oznakama 1.2 in 1.4 predstavljata koncepta prekrška oziroma osebne napake v obrambi. Koncepti 1.2, 1.3 in 1.4 so sestavljeni iz enega samega dogodka, zato nadaljnje drobljenje na podkoncepte ni možno. Po drugi strani pa je koncept 1.1 sestavljen iz večih podkonceptov, ki so zaradi večje preglednosti predstavljeni z rdečimi okvirji brez oznak.

4.2.2 *Algoritem za identifikacijo konceptov*

Sistematično identifikacijo konceptov in podkonceptov realiziramo s preprostim rekurzivnim postopkom. Izhajamo iz matrike razdalj D, ki jo izračunamo na podlagi izbranega kriterija podobnosti med osnovnimi stanji. Na podlagi matrike D zgradimo dendrogram H ter ga ustrezno porežemo. Za vsako dobljeno poddrevo $subH$ izvedemo naslednje. Sestavimo množico M, ki vsebuje liste poddrevesa $subH$ in jo dodamo v seznam identificiranih konceptov C. Nato poddrevo $subH$ rekurzivno porežemo ter na ta način identificiramo še podkoncepte v njegovi strukturi. Rekurzija se izteče, ko dobimo poddrevo $subH$ z enim samim listom oziroma, ko poddrevesa ni več možno porezati. Na koncu vrnemo seznam identificiranih konceptov C in zaključimo.

Izvedba opisanega postopka je podana z algoritmom 2, pri čemer namesto rekurzivnih klicev uporabljamo tehniko pregledovanja v globino s pomočjo sklada. Za delo s skladom potrebujemo osnovne funkcije za inicializacijo (*initStack*), dodajanje in odstranjevanje elementov (*push* in *pop*), ter funkcijo za preverjanje, ali je sklad prazen (*isEmpty*). Algoritem na vhodu prejme matriko razdalj D nad neko (pod)množico dogodkov. Izhod algoritma je seznam identificiranih konceptov. Za delo s seznamom potrebujemo funkciji za inicializacijo (*initList*) ter dodajanje elementov (*append*). Ker pri nadaljnih korakih v postopku gradnje atributnega prostora (opisani so v nadaljevanju besedila) potrebujemo enolično označene koncepte in njihovo pozicijo v hierarhiji, je vsak element seznama identificiranih konceptov sestavljen iz treh komponent:

- komponenta *events* predstavlja množico osnovnih dogodkov, ki definirajo koncept,
- komponenta *id* predstavlja enolično oznako koncepta,
- komponenta *parent* predstavlja oznako nadkoncepta (starša) v hierarhiji.

Na začetku inicializiramo števec *id*, ki bo služil za enolično označevanje identificiranih konceptov, pripravimo seznam C za shranjevanje rezultata in sklad S. Elementi na skladu vsebujejo dve komponenti:

- trenutno poddrevo (komponenta *hierarchy*),
- oznako nadkoncepta, ki ustreza starševskem vozlišču korena trenutnega poddrevesa (komponenta *parent*).

Na vrh sklada postavimo element s celotnim dendrogramom, ki ga zgradimo na podlagi podane matrike razdalj D (funkcija *hierarchicalClustering*). Dokler sklad ni prazen, izvajamo naslednjo zanko. Odstranimo vrhnji element sklada s trenutnim poddrevesom H in oznako nadkoncepta *parent*. Sedaj v skladu s postopki, opisanimi na začetku tega razdelka, določimo ustrezno višino rezanja (funkcija $findNumberOfClusters$) ter porežemo hierarhijo H (funkcija *cutHierarchy*). Vsako dobljeno poddrevo *subH* določa nov koncept, ki ga shranimo v seznam C: določimo novo oznako koncepta *id*, s pomočjo funkcije *getLeaves* zberemo množico osnovnih dogodkov v listih in prenesemo oznako *parent* kot oznako nadkoncepta. Če dobljeni koncept ni trivialen (poddrevo *subH* ni identično hierarhiji H in vsebuje vsaj dva dogodka), je tudi njega možno porezati, zato dodamo na sklad element s poddrevesom *subH*, pri čemer je sedaj njegova oznaka *id* v vlogi oznake nadkoncepta.

Algoritem 2

Identifikacija konceptov

Vhod:

matrika razdalj D med stanji $e_i \in \mathscr{S}$.

Izhod:

seznam konceptov C.

```
1:  function IDENTIFYCONCEPTS(D)
2:      id ← 0
3:      C ← initList()
4:      S ← initStack()
5:      node.hierarchy ← hierarchicalClustering(D)
6:      node.parent ← id
7:      push(S, node)

8:      while not isEmpty(S) do
9:          node ← pop(S)
10:         H ← node.hierarchy
11:         parent ← node.parent
12:         n ← findNumberOfClusters(H)
13:
```

Nadaljuje se na naslednji strani

Algoritem 2

Identifikacija konceptov (nadaljevanje)

```
14:          if n > 1 then
15:              F = {subH} ← cutHierarchy(H, n)
16:              for each subH ∈ F do
17:                  id ← id + 1
18:                  concept.events ← getLeaves(subH)
19:                  concept.id ← id
20:                  concept.parent ← parent
21:                  C ← append(C, concept)
22:                  if |concept.events| > 1 then
23:                      node.hierarchy ← subH
24:                      node.parent ← id
25:                      push(S, node)
26:                  end if
27:              end for
28:          end if
29:      end while
30:      return C
31: end function
```

4.3 Konstrukcija atributnega prostora

V nadaljevanju je opisan postopek za avtomatsko konstrukcijo atributnega prostora. Vhod predstavlja množica opazovanih realizacij stohastičnega procesa v obliki zaporedij osnovnih dogodkov iz $\mathscr{S}$. Izhod postopka je seznam konstruiranih atributov, ki so podani v simbolični obliki kot ulomki X/Y, kjer sta X in Y množici in velja $Y \subseteq \mathscr{S}$, $X \subset Y$. Do končnega rezultata pridemo v dveh korakih. V prvem koraku zagradimo definicije (formule) novih atributov, ki predstavljajo kandidate za atributni prostor. V drugem koraku izberemo najbolj ustrezno podmnožico generiranih atributov.

4.3.1 Konstrukcija definicij novih atributov

Ideja postopka za konstrukcijo definicij atributov temelji na abstraktnih konceptih, ki jih poiščemo z algoritmom 2. Na podlagi podobnosti verjetnostne porazdelitve pred-

nika identificiramo koncepte v celotnem prostoru osnovnih dogodkov. Vsak izmed teh konceptov je izhodišče za izpeljavo množice atributov. Atributi, izpeljani iz posameznega koncepta, imajo v svoji definiciji skupni imenovalec Y, sestavljen iz osnovnih dogodkov, ki definirajo ta koncept. Števce X v definicijah teh atributov dobimo iz konceptov, ki jih identificiramo izključno znotraj osnovnih dogodkov iz imenovalca, le da tokrat uporabimo kriterij podobnosti verjetnostne porazdelitve naslednika.

Potek konstrukcije atributov si poglejmo na konkretnem primeru. Na sliki 4.5(a) je prikazan eden izmed konceptov, ki jih dobimo na podlagi podobnosti verjetnostne porazdelitve prednika. Konkretno gre za koncept žive žoge, ki ni v posesti nobene ekipe. Ta situacija, do katere pride po zgrešenem domačem metu iz igre, ustreza stanju $Pogoj_Y$ na sliki 4.1. Množica dogodkov {ADRB, ADREB, HORB, HOREB} predstavlja imenovalec v definicijah novih atributov, ki jih izpeljujemo na podlagi tega koncepta. Če sedaj dogodke iz opisane množice grupiramo po podobnosti verjetnostne porazdelitve naslednika, dobimo različne načine nadaljevanja situacije, ki je opisana s konceptom v imenovalcu. Te situacije ustrezajo stanjem $Posledica_{X_i}$ na sliki 4.1. Iz dobljenega rezultata, prikazanega na sliki 4.5(b), vidimo, da obstajata dva glavna načina za razplet. Modri okvir z oznako 1 predstavlja koncept skoka v napadu, ki ga izvede domača ekipa. Modri okvir z oznako 2 predstavlja koncept skoka v obrambi s strani gostujoče ekipe. Oba koncepta vsebujeta dva podkoncepta, glede na to, ali je skok pripisan igralcu ali se šteje kot timski skok. Identificirani koncepti na sliki 4.5(b) določajo števce v definicijah novih atributov. Tabela 4.3 prikazuje dobljene formule, vizualizacija prvih dveh pa je prikazana na sliki 4.6. Če bi postopek ponovili še za ostale rdeče okvirje s slike 4.2 (ter za vse njihove podkoncepte), bi zgradili celoten nabor atributov za košarko.

Podrobnejši opis postopka za avtomatsko konstrukcijo definicij novega atributnega prostora je podan z algoritmom 3. Vhod v algoritem je učna množica z opazovanimi realizacijami stohastičnega procesa, podanimi kot zaporedja osnovnih dogodkov iz končne množice $\mathscr{S}$. Izhod algoritma je seznam konstruiranih definicij atributov, pri čemer je posamezen element seznama sestavljen iz naslednjih komponent:

- komponenta x predstavlja množico osnovnih dogodkov v števcu,
- komponenta y predstavlja množico osnovnih dogodkov v imenovalcu,
- komponenta $xParent$ predstavlja oznako koncepta v števcu,
- komponenta yId predstavlja oznako koncepta v imenovalcu.

Slika 4.5

a) Koncept žive žoge po zgrešenem metu iz igre s strani domače ekipe. Nobena ekipa nima posesti žoge. b) Koncepti, dobljeni na podlagi kriterija podobnosti verjetnostne porazdelitve naslednika. Posest žoge lahko pridobi bodisi domača ekipa s skokom v napadu bodisi gostujoča ekipa z obrambnim skokom. Na podlagi identificiranih konceptov lahko sestavimo več atributov: dogodki v modrih okvirjih definirajo števce novih atributov, medtem ko so imenovalci sestavljeni iz dogodkov v rdečem okvirju.

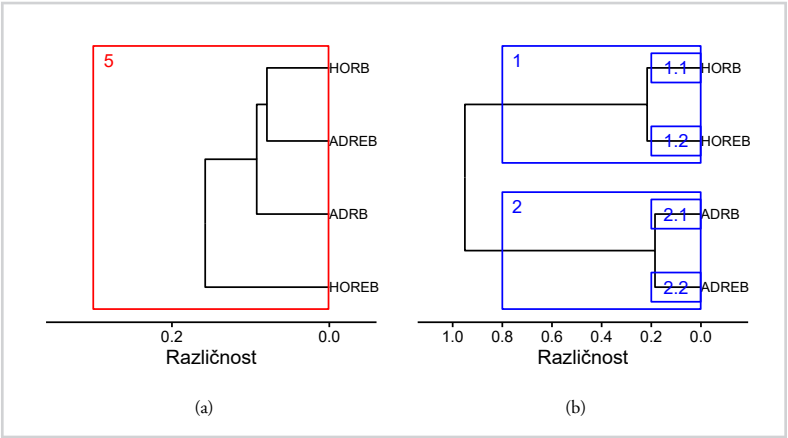
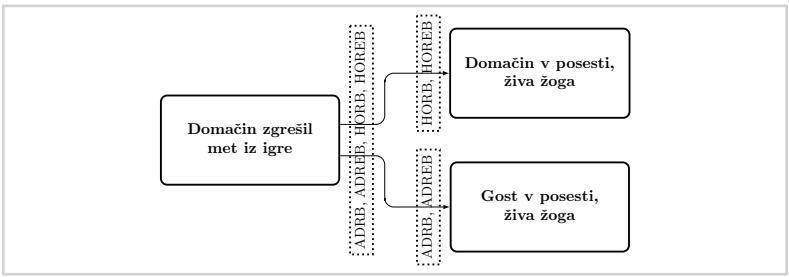


Tabela 4.3

Generirane definicije atributov na podlagi konceptov na sliki 4.5.

{HORB, HOREB} / {ADRB, ADREB, HORB, HOREB}
{ADRB, ADREB} / {ADRB, ADREB, HORB, HOREB}
{HORB} / {ADRB, ADREB, HORB, HOREB}
{HOREB} / {ADRB, ADREB, HORB, HOREB}
{ADRB} / {ADRB, ADREB, HORB, HOREB}
{ADREB} / {ADRB, ADREB, HORB, HOREB}



Slika 4.6

Vizualizacija prvih dveh formul iz tabele 4.3.

Slednji komponenti sta pomembni v postopku izbire podmnožice zgrajenih atributov, ki ga opisujemo v naslednjem razdelku. Za delo s seznamom potrebujemo funkcije za inicializacijo (*initList*) in dodajanje elementov (*append*).

Na začetku postopka iz podanih učnih sekvenc konstruiramo verjetnostni porazdelitvi P_{evt_i} in Q_{evt_i} za vsak osnovni dogodek $evt_i \in \mathscr{S}$ ter na osnovi le-teh izračunamo matriki razdalj D_Q in D_P.

Matriko D_Q uporabimo za identifikacijo konceptov C_Y (s pomočjo algoritma 2). Za vsak element seznama c_Y izvajamo naslednje. Najprej preverimo, ali množica $c_Y.events$ vsebuje vsaj dva dogodka. Ti dogodki sestavljajo imenovalec Y v definiciji novega atributa, ki bi v primeru enega samega člena (poimenujmo ga *evt*) bil neuporaben, saj bi edina možna oblika formule tedaj bila *evt*/*evt*, kar je enako 1. Če imamo v imenovalcu Y vsaj dva dogodka, želimo te dogodke razdeliti na skupine. Najprej sestavimo podmatriko $subD_P$ tako, da iz matrike D_P izberemo vrstice in stolpce, ki ustrezajo elementom iz imenovalca Y (funkcija *getSubMatrix*). Dobljeno podmatriko nato uporabimo za identifikacijo konceptov C_X. Vsak element seznama C_X določa števec X in v primeru, da velja $X \subset Y$ (ne želimo trivialnih atributov oblike Y/Y, kar je enako 1), dobimo definicijo novega atributa. Ko se sprehodimo čez vse elemente iz C_Y in njim ustrezne sezname C_X, imamo zgrajene vse definicije in zaključimo.

4.3.2 Izbira podmnožice generiranih atributov

Število atributov, ki jih generira algoritem 3, je preveliko za praktično uporabo pri postopkih strojnega učenja. Načeloma bi bilo možno izbiro ustrezne podmnožice atributov izvesti z metodo notranje optimizacije (ang. wrapper) ali s pomočjo validacijske množice, toda opisana postopka bi trajala predolgo. Iz tega razloga potrebujemo hitrejši hevristični postopek za izbiro najbolj ustrezne podmnožice generiranih atributov.

Algoritem 3

Konstrukcija definicij atributov

Vhod:

učna množica sekvenc $L = \{seq_i : seq_i = \langle e_{i_1}, e_{i_2}, \ldots, e_{i_k} \rangle, e_{i_j} \in \mathscr{S}\}$.

Izhod:

seznam atributov A.

```
1:  function constructDefinitions(L)
2:      for each e_i ∈ S do
3:          konstruiraj P_{e_i} in Q_{e_j} na podlagi sekvenc iz L
4:      end for
5:      konstruiraj D_P = [d_ij]_{i,j=1...r}, kjer velja d_ij = d_H(P_{e_i}, P_{e_j})
6:      konstruiraj D_Q = [d_ij]_{i,j=1...r}, kjer velja d_ij = d_H(Q_{e_i}, Q_{e_j})
7:      A ← initList()
8:      C_Y ← identifyConcepts(D_Q)
9:      for each c_Y ∈ C_Y do
10:         if |c_Y.events| > 1 then
11:             subD_P ← getSubMatrix(D_P, c_Y.events)
12:             C_X ← identifyConcepts(subD_P)
13:             for each c_X ∈ C_X do
14:                 if |c_X.events| < |c_Y.events| then
15:                     attr.x ← c_X.events
16:                     attr.y ← c_Y.events
17:                     attr.yId ← c_Y.id
18:                     attr.xParent ← c_X.parent
19:                     A ← append(A, attr)
20:                 end if
21:             end for
22:         end if
23:     end for
24:     return A
25: end function
```

Pri izbiri upoštevamo naslednje kriterije:

- splošnost - želimo atribute, ki se nanašajo na čim bolj pogoste dogodke med potekom tekme,
- maksimizacija entropije - želimo atribute s čim večjo entropijo,
- komplementarnost - želimo atribute, ki pokrivajo čim bolj različne aspekte razvoja tekme.

Splošnost atributa merimo glede na podporo (ang. support) imenovalca Y. Definiramo ga kot delež tranzicij v množici učnih sekvenc L (vhod v algoritem 3), ki jih pokrivajo osnovni dogodki iz Y. Večje pokritje imenovalca sugerira, da se opazovani atribut nanaša na večji del problemskega prostora, in ga kot takega favoriziramo.

Entropijo atributa določimo na podlagi verjetnosti, da bo transakcija iz L pokrita s števcem X ob pogoju, da je pokrita z imenovalcem Y. Če omenjeno verjetnost označimo s p, bo entropija, izražena kot $-p \cdot log_2 p - (1-p) \cdot log_2(1-p)$, imela maksimalno vrednost pri $p = 0.5$. Po drugi strani za vrednosti p blizu 0 oziroma 1 se bo entropija približevala 0, kar sugerira, da je atribut trivialen in za strojno učenje neuporaben. Želimo atribute, ki čim bolj enakomerno razdelijo množico dogodkov, na katere se nanašajo. Iz tega razloga favoriziramo atribute z večjo entropijo.

Komplementarnost atributov v izbrani podmnožici dosežemo s tem, da onemogočimo prekrivanje dogodkov v števcu med različnimi atributi. Iz tega razloga ne bi istočasno izbrali prvega in tretjega atributa iz tabele 4.3, saj si delita osnovni dogodek v števcu (ADRB) in se (vsaj delno) nanašata na isti aspekt razvoja tekme. Poleg tega je zaželjeno izogniti se izbiri kolinearnih atributov (takšnih, da je vrednost enega atributa linearna kombinacija vrednosti ostalih atributov). Na primer, če izberemo prvi atribut iz tabele 4.3, potem drugega ne potrebujemo, saj je vsota njunih vrednosti vedno enaka 1.

Postopek za izbiro podmnožice atributov je podan z algoritmom 4. Vhod v algoritem je množica učnih sekvenc L, seznam konstruiranih atributov A (rezultat algoritma 3) in minimalna zahtevana podpora $minSupport$ izbranih atributov. Izhod algoritma je seznam parov $\langle X, Y \rangle$, ki jih interpretiramo kot definicije atributov oblike X/Y.

Najprej razširimo vse elemente v vhodnem seznamu A tako, da poleg množice dogodkov v števcu in imenovalcu oznake starša koncepta v števcu in oznake koncepta v imenovalcu vsebujejo še:

- komponento *entropy*, ki predstavlja vrednost entropije opazovanega atributa,
- komponento *support*, ki predstavlja vrednost podpore opazovanega atributa,
- komponento *covered*, ki predstavlja zaznamek, ali je opazovani atribut že obdelan ali ne.

Ustrezno postavimo vrednosti dodanim komponentam, pri čemer so vsi atributi na začetku neobdelani (vrednost komponente *covered* postavimo na FALSE).

Atribute izbiramo glede na entropijo, zato uredimo elemente seznama A padajoče glede na vrednost komponente *entropy*. Za komplementarnost izbranih atributov poskrbimo s pomočjo množice E, v kateri hranimo osnovne dogodke, pokrite s števci trenutno izbranih atributov. Splošnost izbranih atributov bo zagotovljena z eliminacijo vseh atributov, ki ne zadovoljujejo minimalne podpore.

Sprehodimo se po urejenem seznamu A in za vsak element izvedemo naslednje. Če atribut zagotavlja minimalno podporo, v števcu ne vsebuje že pokritih dogodkov in je še vedno neobdelan, ga izberemo: dodamo definicijo v izhodni seznam F, dodamo pokrite osnovne dogodke v množico E ter ga označimo za obdelanega (zaznamek *covered* postavimo na TRUE). Sedaj preverimo, ali je v tem trenutku v seznamu A ostal samo še eden neobdelan atribut, ki si z ravnokar dodanim atributom deli oznako koncepta v imenovalcu (vrednost komponente *yId*) in oznako starša koncepta v števcu (vrednost komponente *xParent*). Ta atribut je nepotreben, saj predstavlja linearno kombinacijo že dodanih atributov (ker je ostal edini, je njegov števec komplement unije števcev že dodanih "bratov" od imenovalca). Če tak atribut obstaja, ga označimo za obdelanega, osnovne dogodke iz števca dodamo v množico E, same definicije atributa pa *ne dodamo* v izhodni seznam F. Ko gremo čez zadnji element seznama, imamo v F izbrano podmnožico atributov in zaključimo.

4.4 Primer generiranja atributov za košarko

Kot primer si oglejmo rezultat konstrukcije atributnega prostora za košarko. Ponovno izhajamo iz učne množice, sestavljene iz 3690 tekem rednega dela sezon 2008/2009, 2009/2010 in 2010/2011 lige NBA. Množica osnovnih dogodkov je tista iz tabele 3.1.

S pomočjo algoritma za konstrukcijo definicij (glej algoritem 3) dobimo 88 formul. Pred uporabo algoritma za izbiro najbolj ustrezne podmnožice atributov (glej algoritem 4) je potrebno izbrati vrednost parametra *minSupport*, ki določa prag za rezanje

Algoritem 4

Izbira podmnožice atributov

Vhod:

učna množica sekvenc $L = \{seq_i : seq_i = \langle e_{i_1}, e_{i_2}, \ldots, e_{i_k} \rangle, e_{i_j} \in \mathscr{S}\}$,
seznam definicij A (rezultat funkcije *constructDefinitions* iz algoritma 3),
spodnji prag podpore *minSupport*.

Izhod:

seznam izbranih formul $F = \{F_i = \langle X_i, Y_i \rangle\}$, kjer je $F_i = \frac{X_i}{Y_i}$.

```
1:  function SELECTATTRIBUTES(L, A, minSupport)
2:      for each attr ∈ A do
3:          p = freq(attr.x,L) / freq(attr.y,L)
4:          attr.entropy ← −p · log(p) − (1 − p) · log(1 − p)
5:          attr.support ← freq(attr.y, L)
6:          attr.covered ← FALSE
7:      end for

8:      A ← sort(A)                          ▷ uredimo padajoče glede na entropy

9:      F ← initList()
10:     E ← ∅                                ▷ množica pokritih dogodkov
11:     for each attr ∈ A do
12:         if attr.support ≥ minSupport and
                not attr.covered and
                attr.x ∩ E = ∅ then
13:             F ← append(F, ⟨attr.x, attr.y⟩)
14:             E ← E ∪ attr.x
15:             attr.covered ← TRUE
16:
```

Nadaljuje se na naslednji strani

Algoritem 4

Izbira podmnožice atributov (nadaljevanje)

```
17:                 if ∃!a ∈ A :                          ▷ ali eksistira natanko eden?
                        a.covered = FALSE and
                        a.yId = attr.yId and
                        a.xParent = attr.xParent then
18:                         E ← E ∪ a.x
19:                         a.covered ← TRUE
20:                 end if
21:             end if
22:         end for
23:         return F
24: end function
```

definicij s prenizko podporo. Slika 4.7 kaže vrednosti podpore posameznih formul in nekaj možnih pragov za rezanje. Previsoko postavljen prag bo odrezal definicije, ki se nanašajo na manj pogoste, vendar za razvoj tekme še vedno pomembne dogodke. Po drugi strani pa prenizek prag dopušča izbiro manj splošnih definicij, ki lahko prinesejo šum v atributni prostor in s tem otežijo učenje. Ustrezno vrednost parametra lahko določimo ročno, z uporabo domenskega predznanja, ali pa avtomatsko, na podlagi nekega kriterija (npr. točnost napovedovanja zmagovalca tekme s strani modelov, naučenih na dobljenem atributnem prostoru).

Rezultati, predstavljeni v nadaljevanju, so dobljeni z nastavitvijo $minSupport = 0.025$. V tabeli 4.4 so prikazane konstruirane definicije z minimalno zahtevano podporo, padajoče urejene po entropiji.

Tabela 4.4

Generirane definicije atributov za košarko s podporo nad 0.025. Stolpec *x/y* predstavlja formulo za izračun vrednosti atributa. Vrednost entropije generiranih formul je zaokrožena na dve decimalki. Oznake konceptov v imenovalcu so podane v stolpcu *yId*. Oznake starševskih konceptov za števce so podane v stolpcu *xParent*. Oštevilčevanje konceptov v imenovalcu je globalno, v števcu pa lokalno (za vsak imenovalec začne znova). Oznake konceptov so spremenjene tako, da je razvidna pozicija koncepta v hierarhiji (oznake podkonceptov so ločene s piko). Ena sama pika v stolpcu *xParent* označuje, da je koncept v števcu na globini 1, kar pomeni, da je brez starševskega koncepta.

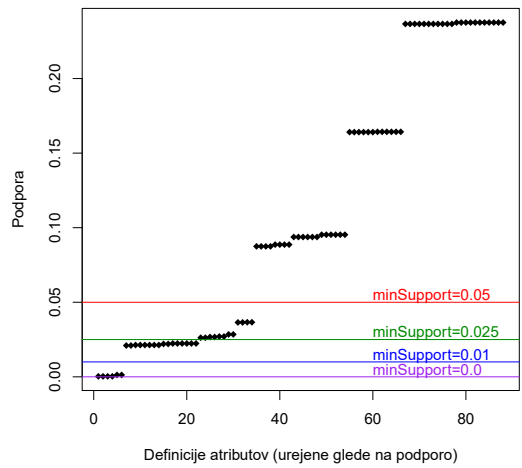
x/y	*entropija*	*yId*	*xParent*
{ H2PA, H3PA } / { H2PA, H2PM, H3PA, H3PM }	1.00	3.3	.
{ H2PM, H3PM } / { H2PA, H2PM, H3PA, H3PM }	1.00	3.3	.
{ A2PA, A3PA } / { A2PA, A2PM, A3PA, A3PM }	0.99	1.1	.

Nadaljuje se na naslednji strani

Tabela 4.4 – nadaljevanje s prejšnje strani

x/y	*entropija*	*yId*	*xParent*
{ A2PM, A3PM } / { A2PA, A2PM, A3PA, A3PM }	0.99	1.1	.
{ H2PM, H3PM, HTO } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.99	3	.
{ A2PM, A3PM, ATO } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.99	1	.
{ A2PA } / { A2PA, A2PM, A3PA, A3PM }	0.97	1.1	1
{ H2PA } / { H2PA, H2PM, H3PA, H3PM }	0.97	3.3	1
{ H2PM } / { H2PA, H2PM, H3PA, H3PM }	0.96	3.3	2
{ A2PA, A3PA } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.96	1	.
{ A2PM } / { A2PA, A2PM, A3PA, A3PM }	0.95	1.1	2
{ H2PA, H3PA } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.95	3	.
{ H3PA } / { H3PA, H3PM }	0.94	3.3.3	.
{ H3PM } / { H3PA, H3PM }	0.94	3.3.3	.
{ A3PA } / { A3PA, A3PM }	0.94	1.1.3	.
{ A3PM } / { A3PA, A3PM }	0.94	1.1.3	.
{ ADRB } / { ADRB, ADREB, HORB, HOREB }	0.94	2	1
{ HDRB } / { AORB, AOREB, HDRB, HDREB }	0.93	8	2
{ H2PM, H3PM } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.91	3	3
{ A2PM, A3PM } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.90	1	2
{ ADRB, ADREB } / { ADRB, ADREB, HORB, HOREB }	0.89	2	.
{ HORB, HOREB } / { ADRB, ADREB, HORB, HOREB }	0.89	2	.
{ ADRB } / { ADRB, ADREB, HORB }	0.89	2.1	1
{ AORB, AOREB } / { AORB, AOREB, HDRB, HDREB }	0.88	8	.
{ HDRB, HDREB } / { AORB, AOREB, HDRB, HDREB }	0.88	8	.
{ HDRB } / { AORB, HDRB, HDREB }	0.88	8.1	2
{ A2PA } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.85	1	1
{ H2PA } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.84	3	2
{ H2PM } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.84	3	3.1
{ A2PM } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.83	1	2.1
{ ADRB, ADREB } / { ADRB, ADREB, HORB }	0.82	2.1	.
{ HORB } / { ADRB, ADREB, HORB }	0.82	2.1	.
{ AORB } / { AORB, HDRB, HDREB }	0.81	8.1	.
{ HDRB, HDREB } / { AORB, HDRB, HDREB }	0.81	8.1	.
{ HORB } / { ADRB, ADREB, HORB, HOREB }	0.79	2	2
{ AORB } / { AORB, AOREB, HDRB, HDREB }	0.78	8	1
{ HFT1A } / { HFT1A, HFT1M }	0.76	9	.
{ HFT1M } / { HFT1A, HFT1M }	0.76	9	.
{ AFT1A } / { AFT1A, AFT1M }	0.76	4	.
{ AFT1M } / { AFT1A, AFT1M }	0.76	4	.
{ AF, AV } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.70	3	.
{ AF } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.70	3	1
{ HF, HV } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.69	1	.
{ HF } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.68	1	3
{ HDREB } / { AORB, HDREB }	0.65	8.1.1	.
{ AORB } / { AORB, HDREB }	0.65	8.1.1	.
{ HORB } / { ADREB, HORB }	0.63	2.1.2	.
{ ADREB } / { ADREB, HORB }	0.63	2.1.2	.
{ A3PA } / { A2PA, A2PM, A3PA, A3PM }	0.59	1.1	1
{ H3PA } / { H2PA, H2PM, H3PA, H3PM }	0.59	3.3	1
{ ATO } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.54	1	2
{ HTO } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.52	3	3
{ A3PA } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.47	1	1
{ H3PA } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.46	3	2
{ H3PM } / { H2PA, H2PM, H3PA, H3PM }	0.40	3.3	2
{ A3PM } / { A2PA, A2PM, A3PA, A3PM }	0.40	1.1	2
{ AOREB } / { AORB, AOREB, HDRB, HDREB }	0.37	8	1
{ HOREB } / { ADRB, ADREB, HORB, HOREB }	0.35	2	2
{ H3PM } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.31	3	3.1
{ A3PM } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.31	1	2.1
{ HDREB } / { AORB, HDRB, HDREB }	0.28	8.1	2
{ ADREB } / { ADRB, ADREB, HORB }	0.28	2.1	1
{ HDREB } / { AORB, AOREB, HDRB, HDREB }	0.27	8	2
{ ADREB } / { ADRB, ADREB, HORB, HOREB }	0.26	2	1
{ AV } / { AF, AV, H2PA, H2PM, H3PA, H3PM, HTO }	0.03	3	1
{ HV } / { A2PA, A2PM, A3PA, A3PM, ATO, HF, HV }	0.03	1	3

Algoritem za izbiro podmnožice atributov efektivno pregleduje definicije v vrstnem redu, kot so izpisane po vrsticah v tabeli 4.4. Najprej doda definicijo iz prve vrstice v seznam izbranih atributov, hkrati pa doda dogodka H2PA in H3PA v množico pokritih



Slika 4.7

Podpora generiranih definicij.

osnovnih dogodkov. Ker je definicija iz druge vrstice sedaj ostala edini "brat" dodane definicije iz prve vrstice (obe imata enako vrednost v stolpcih *yId* in *xParent*, kar pomeni, da je unija števcev obeh definicij enaka imenovalcem in sta zato definiciji kolinearni), je ne bo dodal v seznam izbranih atributov, bo pa dodal dogodka H2PM in H3PM v množico pokritih osnovnih dogodkov. Iz podobnih razlogov bo definicija iz tretje vrstice dodana v izbrano podmnožico atributov, medtem ko bo tista iz četrte vrstice preskočena. Definicija iz pete vrstice ne bo dodana, ker sta dogodka H2PM in H3PM iz števca že pokrita z izbranimi atributi, in tako naprej. Postopek se zaključi, ko pregleda vse generirane definicije. Končna množica izbranih atributov je predstavljena v tabeli 4.5.

Avtomatsko generirani atributi iz tabele 4.5 so zelo podobni (nekateri so celo identični) uveljavljenim statistikam za opis zmogljivosti košarkarske ekipe (glej tabelo 3.2), ki so dobljene z ekspertnim znanjem. Če se postavimo v perspektivo domačega moštva, je prvi avtomatsko generirani atribut iz zgornje tabele ustreznica ekspertnemu atributu *EFG*%, s to razliko, da ne utežuje uspešnih metov za tri točke. Drugi avtomatsko generirani atribut ustreza ekspertnemu atributu *oEFG*%. Avtomatski atribut iz pete

Tabela 4.5

Izbrane definicije za vrednost parametra $minSupport$ = 0.025.

{H2PA, H3PA} / {H2PA, H2PM, H3PA, H3PM}
{A2PA, A3PA} / {A2PA, A2PM, A3PA, A3PM}
{ADRB} / {ADRB, ADREB, HORB, HOREB}
{HDRB} / {AORB, AOREB, HDRB, HDREB}
{HORB, HOREB} / {ADRB, ADREB, HORB, HOREB}
{AORB, AOREB} / {AORB, AOREB, HDRB, HDREB}
{HFT1A} / {HFT1A, HFT1M}
{AFT1A} / {AFT1A, AFT1M}
{AF, AV} / {AF, AV, H2PA, H2PM, H3PA, H3PM, HTO}
{HF, HV} / {A2PA, A2PM, A3PA, A3PM, ATO, HF, HV}
{ATO} / {A2PA, A2PM, A3PA, A3PM, ATO, HF, HV}
{HTO} / {AF, AV, H2PA, H2PM, H3PA, H3PM, HTO}

vrstice je identičen ekspertnemu atributu $ORBr$. Avtomatski atribut iz četrte vrstice je praktično enak ekspertnemu atributu $DRBr$ (v številu manjka dogodek za ekipni skok v obrambi HDREB, ki je relativno redek in ne vpliva bistveno na vrednost atributa). Avtomatski atribut iz sedme vrstice po strukturi spominja na ekspertni atribut $FTF\%$, s to razliko, da generirana formula upošteva samo uspešnost izvajanja zadnjega prostega meta v seriji. Enako velja za avtomatski atribut iz osme vrstice, ki je podoben ekspertnemu atributu $oFTF\%$. Avtomatski atribut v zadnji vrstici je podoben ekspertnemu atributu $TOVr$. V imenovalcu ekspertne formule je člen $0.44 * FTA$, kar je ocena števila posesti, ki so se končale s prekrškom za proste mete. V avtomatsko generirani formuli namesto tega člena nastopata AF (število osebnih napak gostov) in AV (število prekrškov gostov). Formuli bi bili praktično enakovredni, če bi vsaka osebna napaka oziroma prekršek pomenil tudi izvajanje prostih metov.

5

Analiza primera:Modeliranje poteka košarkarske tekme

Za modeliranje poteka košarkarske tekme med specifičnima nasprotnikoma smo uporabili podatke play-by-play tekem, odigranih tekom rednega dela osmih sezon lige NBA (od 2008/2009 do 2015/2016). Vse sezone vsebujejo 1230 tekem, razen sezon 2011/12 (990 tekem) in 2012/13 (1229 tekem). Podatke smo pridobili na spletnem naslovu `stats.nba.com`. Primer zapisa o poteku dogodkov na tekmi je prikazan v tabeli 5.1.

Tabela 5.1

Izsek podatkov play-by-play iz prve četrtine NBA tekme, odigrane 28. marca 2010 med gostujočim Denverjem in domačim Orlandom. En zapis vsebuje podatek o četrtini (stolpec Qtr), času do konca četrtine (stolpec Time), ekipi (stolpec Team) in igralcu (stolpec Player), ki je izvedel akcijo (stolpec Action). Stolpec Score vsebuje oznako vodilne ekipe in trenutni rezultat.

Qtr	Time	Team	Player	Action	Score
1	7:37	DEN	C Billups	Defensive Rebound	DEN 8-8
1	7:21	DEN	N Hilario	2 Point Field Goal	DEN 10-8
1	7:05	ORL	M Barnes	2 Point Miss	DEN 10-8
1	7:04	DEN	N Hilario	Defensive Rebound	DEN 10-8
1	6:54	DEN	J Petro	2 Point Miss	DEN 10-8
1	6:54	DEN	Team	Offensive Rebound	DEN 10-8
1	6:51	DEN	N Hilario	2 Point Miss	DEN 10-8
1	6:48	DEN	A Afflalo	Offensive Rebound	DEN 10-8
1	6:40	ORL	D Howard	Foul	DEN 10-8
1	6:40	DEN	N Hilario	1 Point Free Throw	DEN 11-8
1	6:40	DEN	N Hilario	Missed Free Throw	DEN 11-8
1	6:38	ORL	M Barnes	Defensive Rebound	DEN 11-8

5.1 Kodiranje prostora stanj

Potek košarkarske tekme predstavlja realizacijo stohastičnega procesa, ki je določen tako s splošnimi zakonitostmi športa kot z zmogljivostmi tekmovalnih ekip. Zaradi privzete markovske lastnosti procesa želimo čim več informacije, relevantne za razvoj tekme, vključiti v opis trenutnega stanja. V razdelku 3.3 smo poudarili pomen lokalnega konteksta na modeliranje razvoja tekme, zato prostor stanj kodiramo v obliki vektorja:

$$\langle Evt, Dur, Qtr, Time, PtsDiff, \mathbf{a}, \mathbf{h} \rangle$$

pri čemer komponenta *Evt* predstavlja trenutni dogodek iz množice osnovnih dogodkov (glej tabelo 3.1), ostale komponente pa opisujejo kontekst, v katerem se je opazovani dogodek zgodil. Komponenta *Dur* predstavlja pretekli igralni čas (v sekundah) od prejšnjega dogodka. Trenutek, ko je nastopil dogodek *Evt*, je predstavljen s četrtino (komponenta *Qtr*) in preostalim časom do konca četrtine (komponenta *Time*). Komponenta *PtsDiff* predstavlja rezultat tekme (razlika v koših iz perspektive domačega moštva), ki je veljal neposredno pred dogodkom *Evt*. Končno, vektorja $\mathbf{a}$ in $\mathbf{b}$ predstavljata opis zmogljivosti gostujoče oziroma domače ekipe. Slednja vektorja izrazimo v prostoru ekspertnih atributov (če je le-ta na voljo), oziroma konstruiramo atributni prostor s postopkom, opisanim v razdelku 4.3.

Tukaj je potrebno opozoriti na redundantnost komponent *Dur*, *Qtr* in *Time* (zadoščali bi samo slednji komponenti, medtem ko je vrednost *Dur* možno rekonstruirati iz opisa dveh sosednjih stanj[1]), ki je vendarle praktična - ponuja samozadostnost opisa stanja.

Prehod iz trenutnega stanja $\mathbf{x}$ v naslednje stanje $\mathbf{y}$ je opredeljen z dvema slučajnima spremenljivkama: *Evt*, ki določa osnovni dogodek v opisu naslednjega stanja, in *Dur*, ki določa trajanje prehoda, medtem ko so vse ostale komponente bodisi logična posledica teh spremenljivk (*Qtr*, *Time* in *PtsDiff*) bodisi ostajajo nespremenjene celotno tekmo (karakteristike ekip $\mathbf{a}$ in $\mathbf{b}$). V skladu z izrazom 3.3 lahko pogojno porazdelitev $f(\mathbf{y}|\mathbf{x})$ predstavimo v obliki:

$$f(\mathbf{y}|\mathbf{x}) = f_{\mathbf{Evt}}(y^{(Evt)}|\mathbf{x}) f_{\mathbf{Dur}}(y^{(Dur)}|\mathbf{x}, y^{(Evt)}) \tag{5.1}$$

Pogojno porazdelitev iz enačbe 5.1 lahko ocenimo s pomočjo dveh modelov: model M_{Evt} ocenjuje marginalno pogojno porazdelitev naslednjega dogodka ob podanem opisu trenutnega stanja, in model M_{Dur}, ki ocenjuje čas med dvema dogodkoma glede na opis trenutnega stanja in naslednji dogodek.

5.2 Napovedovanje naslednjega dogodka

Napovedovanje naslednjega dogodka je klasifikacijski problem in bi ga v principu lahko izvedli s poljubnim parametričnim ali neparametričnim modelom. Pri tem je potrebno upoštevati, da se razvoj košarkarske tekme podreja množici diskretnih pravil igre, ki

[1] Na prvi pogled zadošča samo komponenta *Dur*, a bi s tem izgubili možnost začetka simulacije v poljubnem trenutku.

jih napovedi modela ne smejo kršiti. Po teh pravilih vsakemu dogodku lahko sledi natančno določena podmnožica dogodkov, kar vodi k naravni razdelitvi prostora stanj v disjunktne podmnožice, glede na možnost nadaljevanja razvoja tekme. Omenjeno razbitje prostora lahko izvedemo avtomatsko ali s pomočjo domenskega predznanja, dobljene podprostore pa nato ločeno modeliramo.

Odločitveno drevo z multinomialno logistično regresijo v listih je ustrezna izbira za model M_{Evt} [26]. Dodatna prednost tega modela je neposredna interpretabilnost njegovih odločitev.

5.2.1 *Generiranje atributnega prostora za M_{Evt}*

Avtomatsko konstrukcijo značilk za modeliranje naslednjega dogodka smo izvedli s pomočjo algoritma 3. Učne sekvence za gradnjo atributnega prostora smo dobili s prevedbo podatkov play-by-play v zaporedja z elementi iz ciljne množice osnovnih dogodkov (glej tabelo 3.1). Ena tekma nam poda štiri učne sekvence (po eno na vsako četrtino, podaljške smo ignorirali).

Za primer si poglejmo učno sekvenco, dobljeno na podlagi prvih petih zapisov iz tabele 5.1:

$$\text{ADRB} \rightarrow \text{A2PM} \rightarrow \text{HINB} \rightarrow \text{H2PA} \rightarrow \text{ADRB} \rightarrow \text{A2PA}$$

Iz zgornjega primera je razvidno, da učne sekvence lahko vsebujejo tudi tranzicije, ki niso eksplicitno navedene v podatkih play-by-play, vendar predstavljajo logično posledico prejšnjega dogodka (v konkretnem primeru dogodek HINB sledi dogodku A2PM, saj je gostujoča ekipa dosegla 2 točki iz igre in domačin pridobi posest žoge).

Množico učnih sekvenc smo sestavili na podlagi 3690 tekem iz treh NBA sezon (od 2008/2009 do 2010/2011). Z različnimi nastavitvami parametra za minimalno podporo (glej sliko 4.7) smo pridobili štiri skupine atributov, ki so predstavljene v tabeli 5.2.

5.2.2 *Priprava učne množice za M_{Evt}*

Učno množico za gradnjo modela M_{Evt} sestavimo tako, da podatke play-by-play prevedemo v zaporedje prehodov v izbranem prostoru stanj. En učni primer predstavlja tranzicijo med zaporednima dogodkoma. Vsaki komponenti vektorja za kodiranje

Tabela 5.2

Skupine atributov za modeliranje naslednjega dogodka, dobljene z različnimi nastavitvami parametra za minimalno podporo v algoritmu 4. Za primerjavo je podana tudi skupina ekspertnih atributov z oznako FF, ki ustrezajo definicijam iz tabele 3.2.

Oznaka skupine	Minimalna podpora	Število atributov
ACA0	-	17
ACA10	0.01	16
ACA25	0.025	12
ACA50	0.05	10
FF	-	8

stanj ustreza en atribut (stolpec) v učni množici, razen komponentam za opis zmogljivosti ekip, ki sta sama vektorja in se raztezata čez več stolpcev (vsak element teh vektorjev predstavlja svoj atribut).

Pri izdelavi učnih sekvenc je potrebno upoštevati, da zapisi play-by-play vsebujejo določeno število napak, ki jih je priporočljivo prečistiti pred nadaljevanjem. Za avtomatsko odstranjevanje neveljavnih prehodov v dobljenih sekvencah smo uporabili naslednjo hevristiko: prehod iz osnovnega stanja e_i v osnovno stanje e_j obdržimo, če je vsaj ena izmed vrednosti $P_{e_i}(e_j)$ ali $Q_{e_j}(e_i)$ nad nekim vnaprej določenim pragom. Drugače povedano, prehod obdržimo, če je bodisi stanje e_j zadosti pogost naslednik stanja e_i bodisi je stanje e_i zadosti pogost predhodnik stanja e_j. V naših eksperimentih smo uporabili vrednost praga 0.01. Opisana hevristika bo odstranila tudi veljavne prehode, ki se razmeroma redko zgodijo. Kot taki malo prispevajo h generalizaciji razvoja dogodkov na tekmi, lahko pa pripeljejo do prekomernega prileganja podatkom in jih je zato priporočljivo odstraniti iz učne množice.

Vrednosti atributov za opisovanje ekip določimo za vsako tekmo posebej na podlagi štetja prehodov med dogodki v tekmah, ki so odigrane do tega trenutka. Pri izračunu atributov za opisovanje domačega moštva upoštevamo samo tekme, ki jih je ta ekipa odigrala doma (ne glede na nasprotnika). Prav tako pri izračunu atributov za gostujočo ekipo upoštevamo samo tekme, ki so jih odigrali v gosteh (ne glede na nasprotnika). Ideja za ločeno obravnavanje domačih in gostujočih tekem je v tem, da na ta način implicitno upoštevamo prednost domačega igrišča. Primer učne množice, ki ustreza podatkom play-by-play iz tabele 5.1, je prikazan v tabeli 5.3.

Tabela 5.3

Učni primeri, ki ustrezajo dogodkom play-by-play iz tabele 5.1. Posamezna vrstica predstavlja tranzicijo in dogodka PrevEvt v Evt (ciljna spremenljivka). Stolpci od Aatt_1 do Aatt_w predstavljajo vrednosti atributov, ki opisujejo razvoj dogodkov na tekmah gostujoče ekipe (lahko jih interpretiramo kot zmogljivosti gostujoče ekipe). Podobno stolpci od Hatt_1 do Hatt_w opisujejo razvoj dogodkov na tekmah domačina, kar lahko interpretiramo kot zmogljivosti domače ekipe. Stolpci Qtr, Time in PtsDiff predstavljajo četrtino, čas do konca četrtine (v sekundah) in trenutno razliko v rezultatu (iz perspektive domače ekipe) ob začetku tranzicije. Stolpec PrevDur je čas trajanja prejšnje tranzicije (tiste, ki je pripeljala do dogodka PrevEvt).

Aatt_1	…	Aatt_w	Hatt_1	…	Hatt_w	Qtr	Time	PtsDiff	PrevDur	PrevEvt	Evt
a_1	…	a_w	h_1	…	h_w	1	457	0	-	ADRB	A2PM
a_1	…	a_w	h_1	…	h_w	1	441	-2	16	A2PM	HINB
a_1	…	a_w	h_1	…	h_w	1	441	-2	0	HINB	H2PA
a_1	…	a_w	h_1	…	h_w	1	425	-2	16	H2PA	ADRB
a_1	…	a_w	h_1	…	h_w	1	424	-2	1	ADRB	A2PA
a_1	…	a_w	h_1	…	h_w	1	414	-2	10	A2PA	AOREB
a_1	…	a_w	h_1	…	h_w	1	414	-2	0	AOREB	A2PA
a_1	…	a_w	h_1	…	h_w	1	411	-2	3	A2PA	AORB
a_1	…	a_w	h_1	…	h_w	1	408	-2	3	AORB	HF
a_1	…	a_w	h_1	…	h_w	1	400	-2	8	HF	AFT2M
a_1	…	a_w	h_1	…	h_w	1	400	-3	0	AFT2M	AFT1A
a_1	…	a_w	h_1	…	h_w	1	400	-3	0	AFT1A	HDRB

5.2.3 *Primerjava različnih modelov za M_{Evt}*

Primerjali smo različne modele po točnosti napovedovanja naslednjega dogodka na košarkarski tekmi.

HOM je najpreprostejši izmed obravnavanih modelov za M_{Evt}. Verjetnostna porazdelitev naslednjega dogodka je modelirana kot relativna frekvenca možnih nadaljevanj, pogojena s trenutnim dogodkom (ostale atribute iz učne množice ignoriramo).

Preostali modeli so v obliki odločitvenega drevesa z multinomialno logistično regresijo v listih. Korensko vozlišče drevesa je ročno izbrano in predstavlja ne-binarno razbitje problemskega prostora glede na trenutni dogodek (predstavlja ga vrednost atributa *PrevEvt*), ustrezna poddrevesa pa so avtomatsko zgrajena z rekurzivno binarno delitvijo, pri čemer se kot mera za izbiro delitvenega kriterija uporablja ocena MDL [59]. Gradnja drevesa se ustavi, ko število primerov v vozlišču pade pod 500, ali ko nobeno razbitje ni kompresivno glede na oceno MDL. Če ni drugače navedeno, logistični modeli v listih drevesa uporabljajo celoten nabor atributov iz učne množice, na podlagi katere je drevo zgrajeno.

QTD je model, naučen na podmnožici atributov, ki se nanašajo samo na lokalni kontekst tekme (četrtina, čas do konca četrtine, razlika v rezultatu, trajanje prejšnje akcije in trenutni dogodek), medtem ko se opisi zmogljivosti ekip ignorirajo.

FF je model, podoben zgornjemu, le s to razliko, da drevo gradimo na podlagi vseh atributov v učni množici, vključno z zmogljivostmi ekip. Slednje so opisane z ekspertnimi atributi iz tabele 3.2.

ACA0, *ACA10*, *ACA25* in *ACA50* so modeli, podobni zgornjemu, s to razliko, da so kot atributi za opisovanje zmogljivosti ekip uporabljene istoimenske skupine iz tabele 5.2.

*ACA0**, *ACA10**, *ACA25** in *ACA50** so prav tako naučeni z atributi iz tabele 5.2. Za razliko od zgoraj opisanih modelov, logistični modeli v listih drevesa uporabljajo samo podmnožico atributov, ki jo dobimo z naslednjo hevristiko. Zaradi dejstva, da korensko vozlišče razdeli prostor glede na trenuten dogodek, posledično za vsako poddrevo natančno vemo, katere možne naslednje dogodke modelira. Hkrati pa zaradi interpretabilnosti generiranih atributov vemo, na katere dogodke se ti nanašajo (na podlagi množice dogodkov v imenovalcu konstruiranih formul). To nam omogoča, da pri učenju multinomialnih logističnih modelov v listih posameznega poddrevesa uporabimo samo tiste atribute, ki se našajo na dogodke, ki jih to poddrevo modelira.

Modele smo učili na tekmah iz dveh zaporednih sezon, testirali pa na naslednji, tretji sezoni. Vrednosti atributov v učnih primerih so izračunane za vsako sezono posebej (tj. zmogljivosti ekip na začetku ene sezone se ne navezujejo na njihovo uspešnost v predhodni sezoni). Za vsak testni primer so modeli vrnili verjetnostno porazdelitev dogodka, ki se bo naslednji zgodil na tekmi. Različnost med vrnjenimi predikcijami in dejanskimi dogodki smo ocenili z Brierjevo mero [60] in oceno Log score. Rezultati so prikazani v tabeli 5.4. Na sliki 5.1 je prikazano spreminjanje povprečne Brierjeve mere v odvisnosti od časa do konca četrtine. Zaradi večje preglednosti so prikazane samo meritve izbranih modelov. Slika A.1 v dodatku kaže detajlni prikaz zadnjih 60 sekund četrtine. Iz slike je razvidno, da se napovedi posameznih modelov razlikujejo predvsem v zadnjih nekaj deset sekund pred koncem vsake četrtine. Na sliki 5.2 je prikazano gibanje povprečne Brierjeve mere po igralnih dneh rednega dela NBA sezone 2015/2016.

Iz priloženih rezultatov in slik lahko sklepamo, da je model *HOM* bistveno slabši od ostalih modelov, ki so po uspešnosti napovedovanja naslednjega dogodka dokaj primerljivi. Modeli z izbiro podmnožice atributov v listih imajo prednost pred modeli, ki uporabljajo poln nabor atributov, predvsem na začetku sezone, ko so statistike ekip še nezanesljive, saj so izračunane na podlagi relativno majhnega števila odigranih tekem. Model *QTD* je presenetljivo dober napovedovalec naslednjega dogodka, kljub temu da ne upošteva značilnosti ekip. Iz tega lahko sklepamo, da lokalni kontekst trenutka bolj vpliva na naslednji dogodek kot značilnosti ekip - to posebej drži za ligo NBA, kjer so razlike med ekipami načrtno dokaj majhne, da bi se spodbujala negotovost tekmovanja.

5.3 Napovedovanje časa med dogodkoma

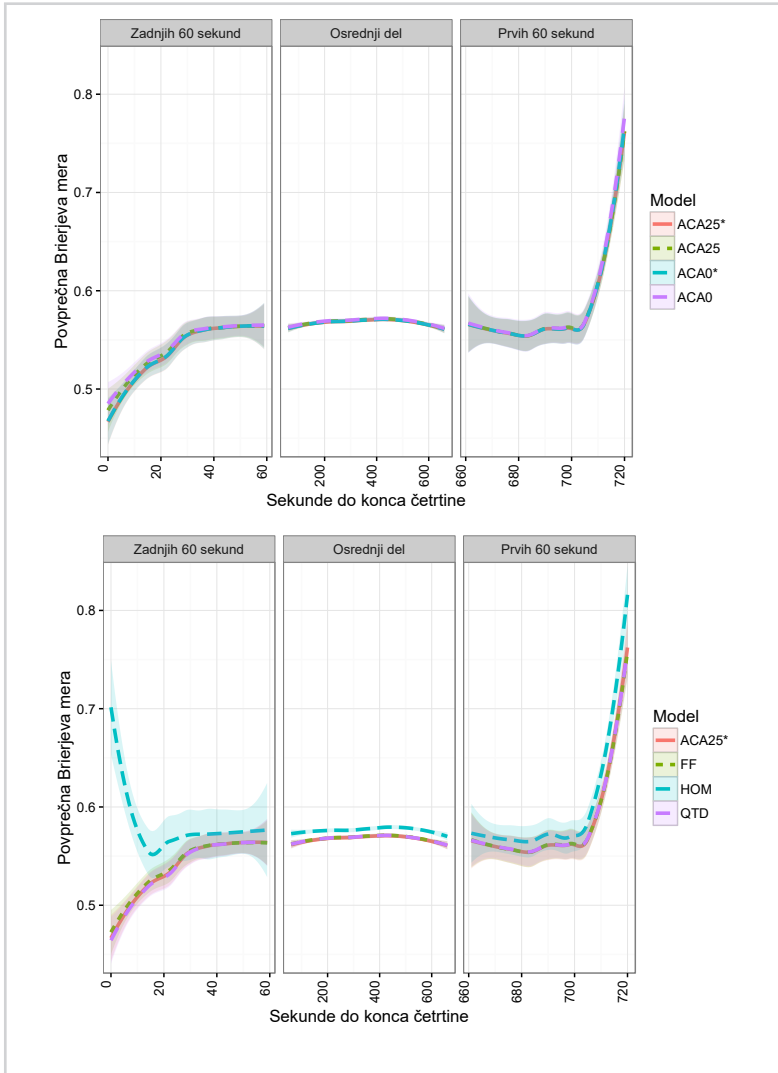
Napovedovanje časa med dvema dogodkoma je v osnovi regresijski problem, saj ciljna spremenljivka lahko zavzame poljubno zvezno vrednost med 0 in 24 (omejitev dolžine napada, izražena v sekundah). Po drugi strani pa lahko čas med dvema dogodkoma obravnavamo tudi kot ordinalno diskretno spremenljivko zaradi vsesplošne sekundne granulacije časa v podatkih play-by-play.

Regresijsko drevo je ustrezen model za M_{Dur} [26], saj hierarhična struktura omogoča naravno delitev problemskega prostora glede na par dogodkov: tistega, ki se nazadnje zgodil, in tistega, ki je napovedan, da se bo naslednji zgodil. Posamezne napovedi o pretoku igralnega časa med omenjenema dogodkoma pa dobimo z vzorčenjem iz

Tabela 5.4

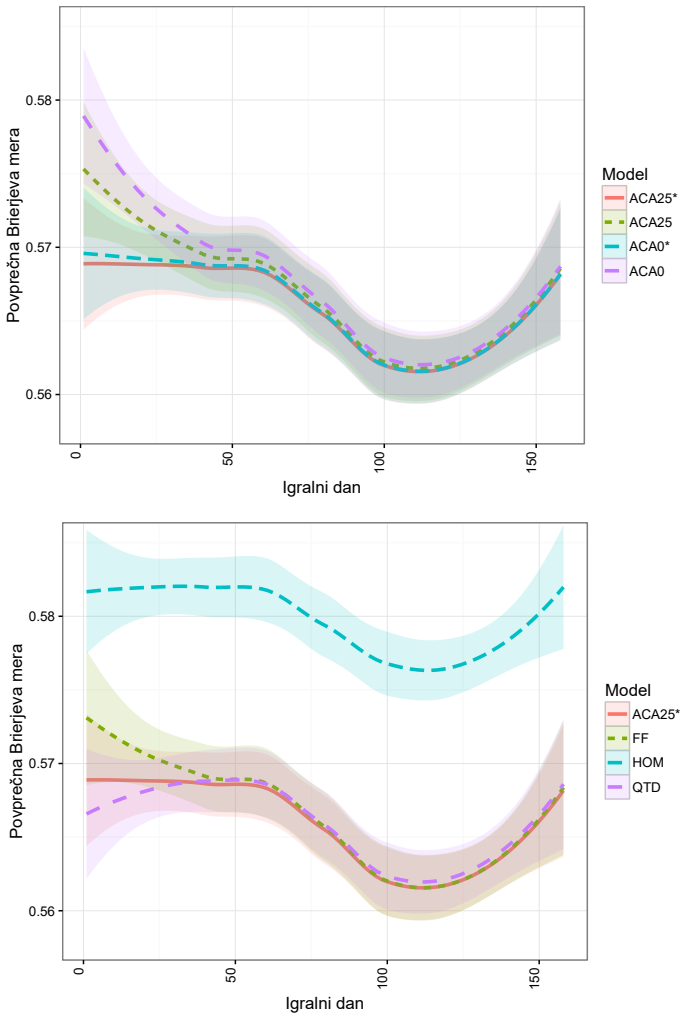
Povprečne vrednosti ocen Log score in Brier score, izmerjene pri napovedovanju naslednjega dogodka v rednem delu tekme (četrtine 1-4) rednega dela NBA sezon 2014/15 in 2015/16. Vrednosti v oklepajih predstavljajo standardno napako.

	2014/15		2015/16	
Model	Log score	Brier score	Log score	Brier score
HOM	1.224 (± 0.0012)	0.5806 (± 0.00055)	1.224 (± 0.0012)	0.5795 (± 0.00055)
QTD	1.187 (± 0.0012)	0.5666 (± 0.00055)	1.188 (± 0.0012)	0.5658 (± 0.00055)
FF	1.189 (± 0.0012)	0.5670 (± 0.00056)	1.190 (± 0.0013)	0.5661 (± 0.00056)
ACA0	1.193 (± 0.0013)	0.5680 (± 0.00056)	1.194 (± 0.0013)	0.5672 (± 0.00056)
ACA10	1.193 (± 0.0013)	0.5683 (± 0.00055)	1.194 (± 0.0013)	0.5671 (± 0.00056)
ACA25	1.190 (± 0.0012)	0.5674 (± 0.00055)	1.192 (± 0.0013)	0.5665 (± 0.00056)
ACA50	1.190 (± 0.0012)	0.5673 (± 0.00055)	1.191 (± 0.0013)	0.5664 (± 0.00056)
*ACA0**	1.188 (± 0.0012)	0.5669 (± 0.00055)	1.189 (± 0.0012)	0.5658 (± 0.00056)
*ACA10**	1.188 (± 0.0012)	0.5668 (± 0.00055)	1.189 (± 0.0012)	0.5657 (± 0.00056)
*ACA25**	1.188 (± 0.0012)	0.5667 (± 0.00055)	1.188 (± 0.0012)	0.5656 (± 0.00056)
*ACA50**	1.188 (± 0.0012)	0.5668 (± 0.00055)	1.188 (± 0.0012)	0.5657 (± 0.00056)
	N = 591397		N = 596345	



Slika 5.1

Povprečna vrednost Brierjeve mere kot funkcije časa do konca četrtine. Ocena se nanaša na verjetnostno napovedovanje naslednjega dogodka na tekmi. Rezultati so prikazani za redni del sezone NBA 2015/16 in so zglajeni po metodi LOESS (faktor glajenja 0.75).



Slika 5.2

Povprečna vrednost Brierjeve mere pri napovedovanju naslednjega dogodka na tekmah rednega dela lige NBA v sezoni 2015/2016. Rezultati so zglajeni po metodi LOESS (faktor glajenja 0.75).

empirične porazdelitve primerov v listih drevesa.

5.3.1 Generiranje atributnega prostora za M_{Dur}

Za učenje modela M_{Dur} lahko uporabimo enak atributni prostor kot pri napovedovanju naslednjega dogodka (tj. učenju modela M_{Evt}), saj značilke za opisovanje nizanja dogodkov implicitno določajo zakonitosti o dinamiki teh dogodkov - posredno opisujejo potek časa med dogodki. Pomanjkljivost tega pristopa je manjša interpretabilnost modela M_{Dur}.

Druga možnost je, da z uporabo algoritma 3 konstruiramo atributni prostor, ki eksplicitno opisuje dinamiko igre. Postopamo podobno kot pri gradnji atributov za napovedovanje naslednjega dogodka - zapise play-by-play prevedemo v zaporedja elementov, ki tokrat označujejo igralni čas med zaporednima dogodkoma. Na primer, na podlagi play-by-play zapisov iz tabele 5.1 dobimo naslednjo učno sekvenco:[2]

$$16 \rightarrow 0 \rightarrow 16 \rightarrow 1 \rightarrow 10 \rightarrow 0 \rightarrow 3 \rightarrow 3 \rightarrow 8 \rightarrow 0 \rightarrow 0 \rightarrow 2$$

Vrednosti v dobljenih sekvencah obravnavamo kot diskretne razrede. Na podlagi množice učnih sekvenc, sestavljene iz istih tekem kot v primeru gradnje atributnega prostora za model M_{Evt}, smo z različnimi nastavitvami parametra za minimalno podporo pridobili tri skupine atributov, ki so pod oznakami ACAS0, ACAS1 in ACAS2 povzete v tabeli 5.5.

Transakcije med zaporednima dogodkoma dolžine 0 so večinoma posledica košarkarskih pravil in ne prispevajo k opisu dinamike igre. Na določen način tudi pokvarijo analizo dinamike igre, saj "presekajo"sekvence, ki ustrezajo dejanski akciji na igrišču in tako vplivajo na porazdelitveni funkciji prednikov in naslednikov. Iz tega razloga smo zgradili še eno družino atributnih prostorov, ki so dobljeni na podlagi učnih sekvenc z odstranjenimi transakcijami dolžine 0. Dobljene skupine atributov, označene kot ACASP0, ACASP2 in ACASP4, so povzete v tabeli 5.5.

Vsi avtomatsko zgrajeni atributi merijo razmerje med krajšimi in daljšimi variantami različno dolgih napadov. Kot primer dobljenih formul so v tabeli 5.6 podani atributi iz skupine ACAS1.

[2] drugi element zaporedja (vrednost 0) ne ustreza eksplicitni vrstici v zapisu play-by-play, temveč označuje takojšen prehod iz doseženega koša v začetek novega napada z vračanjem žoge v igro.

Tabela 5.5

Skupine atributov za modeliranje igralnega časa med zaporednima dogodkoma, dobljene z različnimi nastavitvami parametra za minimalno podporo v algoritmu 4. Skupine ACAS0, ACAS1 in ACAS2 so dobljene na podlagi vseh transakcij iz učnih sekvenc, medtem ko so ACASP0, ACASP2 in ACASP4 dobljene z ignoriranjem transakcij dolžine 0 (takojšnji prehodi, ki so večinoma posledica pravil igre).

Oznaka skupine	Minimalna podpora	Število atributov
ACAS0	-	15
ACAS1	0.1	6
ACAS2	0.2	3
ACASP0	-	11
ACASP2	0.2	6
ACASP4	0.4	2

5.3.2 Priprava učne množice za M_{Dur}

Učna množica za model M_{Dur} je po strukturi skoraj identična tisti za M_{Evt} (glej tabelo 5.3) - razlika je dodatni stolpec *Dur*, ki označuje čas prehoda iz trenutnega dogodka (atribut *PrevEvt*) v napovedan dogodek (atribut *Evt*) in predstavlja ciljno spremenljivko.

Za avtomatsko odstranjevanje morebitnih napak v vhodnih play-by-play podatkih smo uporabili identično hevristiko kot pri pripravi učne množice za model M_{Evt} z dodatnim filtrom, ki odstrani predolge transakcije (omejitev trajanja napada je 24 sekund, kar pomeni, da so vsi daljši prehodi neveljavni).

5.3.3 Primerjava različnih modelov za M_{Dur}

Primerjali smo različne modele po točnosti napovedovanja igralnega časa med zaporednima dogodkoma na tekmi.

HOM je najpreprostejši izmed obravnavanih modelov za M_{Dur}. Čas med zaporednima dogodkoma modelira z vzorčenjem iz populacije vseh prehodov (opaženih v učni množici) med tema dogodkoma.

Ostali modeli so v obliki regresijskega drevesa. Prva dva nivoja drevesne strukture sta ročno določena in razdelita problemski prostor glede na predhodni dogodek (korensko vozlišče) in napovedani naslednji dogodek (vozlišče pod korenom). Preostanek drevesa pa je avtomatsko zgrajen z rekurzivno delitvijo po kriteriju najmanjših kvadratov na podlagi celotnega nabora atributov iz učne množice. Gradnja drevesa se ustavi,

Tabela 5.6

Izbrane definicije za vrednost parametra $minSupport = 0.1$. Številke pomenijo igralni čas v sekundah do naslednjega dogodka. Zgrajene definicije predstavljajo deleže različno dolgih akcij.

$\frac{\{6, 7, 8\}}{\{3, 4, 5, 6, 7, 8\}}$
$\frac{\{15, 16, 17, 18, 19, 20, 21\}}{\{9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24\}}$
$\frac{\{9, 10, 11, 12, 13, 14\}}{\{9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24\}}$
$\frac{\{1\}}{\{1, 2\}}$
$\frac{\{4, 5\}}{\{3, 4, 5, 6, 7, 8\}}$
$\frac{\{0\}}{\{0, 1, 2\}}$

ko v listu ostane premalo učnih primerov, pri čemer se prag eksperimentalno določi na podlagi validacijske množice. Predikcijo modela v obliki diskretne verjetnostne porazdelitve igralnega časa med zaporednima dogodkoma dobimo iz empirične porazdelitve primerov v listih drevesa.

QTD je model, naučen na podmnožici atributov, ki se nanašajo samo na lokalni kontekst tekme (četrtina, čas do konca četrtine, razlika v rezultatu, trajanje prejšnje akcije, trenutni in naslednji dogodek), medtem ko se opisi zmogljivosti ekip ignorirajo.

FF je model, podoben zgornjemu, s to razliko, da se pri gradnji modela upoštevajo tudi atributi za opis zmogljivosti ekip. Uporabljeni so ekspertni atributi iz tabele 3.2.

ACA25 in *ACA50* sta modela, podobna zgornjemu, a so tokrat zmogljivost ekip opisane z istoimensko skupino atributov iz tabele 5.2, ki so sicer namenjeni napovedovanju naslednjega dogodka.

ACAS0, *ACAS1*, *ACAS2*, *ACASP0*, *ACASP2* in *ACASP4* so modeli, podobni zgornjemu, s tem da so tokrat atributi za opisovanje zmogljivosti ekip istoimenske skupine iz tabele 5.5. Uporabljeni atributi so zgrajeni z namenom napovedovanja časa med zaporednima dogodkoma na tekmi.

Za razliko od modela M_{Evt}, ki napoveduje porazdelitev nominalne spremenljivke (naslednji dogodek), model M_{Dur} vrača porazdelitev ordinalne spremenljivke (čas med zaporednima dogodkoma v sekundni natančnosti). Za ocenjevanje kvalitete napovedi posameznih modelov M_{Dur} smo uporabili oceno RPS (Ranked Probability Score)[61], ki primerja vrnjene diskretne verjetnostne porazdelitve z dejanskimi časi na odigranih tekmah:

$$RPS = \frac{1}{r-1} \sum_{i=1}^{r} (\sum_{j=1}^{i} p_j - \sum_{j=1}^{i} e_j)^2 \tag{5.2}$$

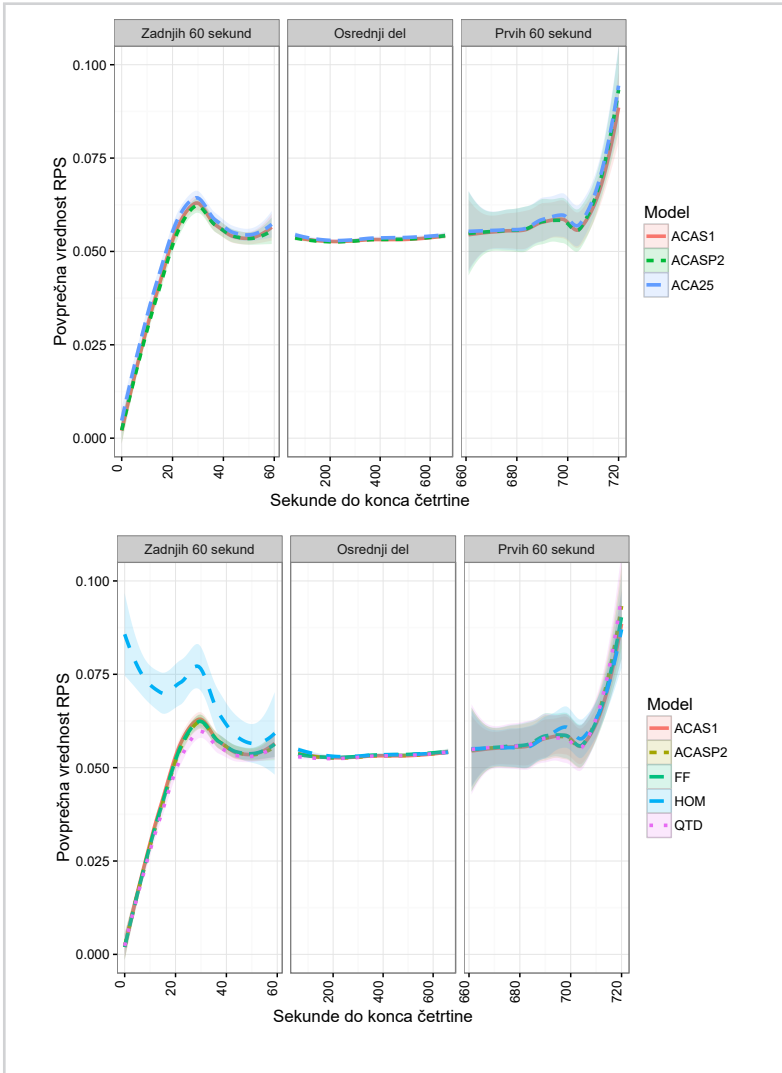
pri čemer r predstavlja število možnih ordinalnih razredov, p_j je napovedana verjetnost j-tega dogodka, e_j pa je dejanski izid j-tega dogodka (vrednost 1 označuje, da se je dogodek zgodil, vrednost 0 pa da se ni).

Vse modele smo učili na podoben način kot pri napovedovanju naslednjega dogodka. Učna množica je bila sestavljena iz tekem dveh zaporednih sezon, naslednja (tretja) sezona pa je bila uporabljena za testiranje kvalitete napovedi. Najprej smo za vsak (drevesni) model določili globino regresijskega drevesa, ki optimizira kvaliteto njegovih napovedi na validacijski množici (redni del sezone 2010/2011). Nato smo ustrezno porezana drevesa testirali na sezonah 2014/2015 (na podlagi učne množice tekem iz 2012/2013 in 2013/2014) in 2015/2016 (na podlagi učne množice tekem iz 2013/2014 in 2014/2015). Rezultati so podani v tabeli 5.7.

Spreminjanje kvalitete napovedi nekaterih modelov M_{Dur} je prikazano na slikah 5.3-5.4. Gibanje ocene RPS v odvisnosti od časa do konca četrtine je prikazano na sliki 5.3. Slika A.2 v dodatku kaže detajlen prikaz zadnjih 60 sekund četrtine. Spreminjanje ocene RPS po igralnih dneh rednega dela NBA sezone 2015/2016 je prikazano na sliki 5.4.

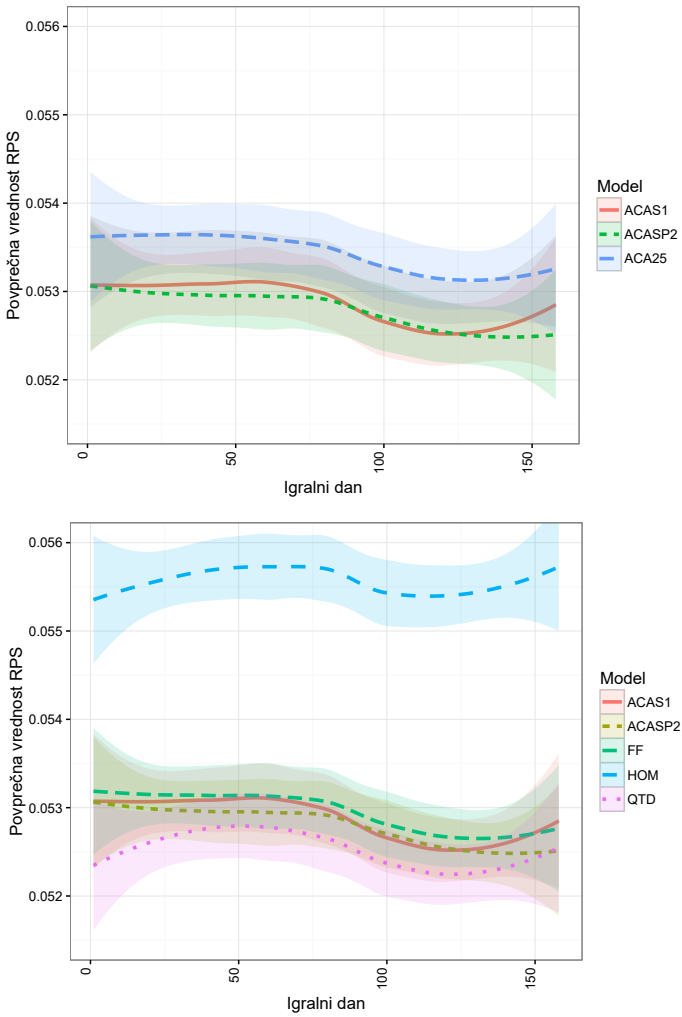
5.3.4 *Analiza verodostojnosti generiranih simulacij*

Za generiranje simulacij poteka košarkarske tekme s pomočjo algoritma 1 potrebujemo par modelov: M_{Evt} za napovedovanje naslednjega dogodka in M_{Dur} za napovedova-



Slika 5.3

Povprečna vrednost ocene RPS kot funkcije časa do konca četrtine. Ocena se nanaša na verjetnostno napovedovanje časa med zaporednima dogodkoma na tekmi. Rezultati so prikazani za redni del sezone NBA 2015/16 in so zglajeni po metodi LOESS (faktor glajenja 0.75).



Slika 5.4

Povprečna vrednost ocene RPS pri napovedovanju časa med zaporednima dogodkoma na tekmah rednega dela lige NBA v sezoni 2015/2016. Rezultati so zglajeni po metodi LOESS (faktor glajenja 0.75).

Tabela 5.7

Povprečne vrednosti ocene RPS za kvaliteto napovedanih porazdelitev časa med zaporednima dogodkoma na košarkarski tekmi, izmerjene na tekmah rednega dela sezon NBA 2014/2015 in 2015/2016. Standardne napake so pri vseh vrednostih enake 0.0001.

	2014/15	2015/16
HOM	0.0556	0.0555
QTD	0.0523	0.0525
FF	0.0528	0.0529
ACA25	0.0534	0.0534
ACA50	0.0534	0.0533
ACAS0	0.0527	0.0527
ACAS1	0.0526	0.0528
ACAS2	0.0526	0.0532
ACASP0	0.0527	0.0528
ACASP2	0.0526	0.0527
ACASP4	0.0526	0.0531
	N = 583880	N = 589045

nje, koliko igralnega časa preteče do nastopa tega dogodka. Od generiranih simulacij pričakujemo, da so v skladu z osnovnimi pravili igre, da sledijo uveljavljenim praksam in racionalnim odločitvam (na primer, ekipa, ki zaostaja za tri točke bo v izdihljajih tekme poskusila izenačiti z metom za tri in ne bo metala za dve točki) ter tudi to, da reflektirajo karakteristike ekip, ki igrajo te navidezne tekme (na primer, za ekipo z dobrimi skakalci pričakujemo, da bo dominirala v igri pod košem).

Generirali smo 10 simulacij za vsako tekmo iz sezon 2014/2015 in 2015/2016. Pri generiranju simulacij tekem iz ene sezone smo uporabili modele, naučene na prejšnjih dveh sezonah. Na ta način smo z vsakim parom modelov M_{Evt}-M_{Dur} dobili 23800 simulacij[3], ki smo jih nato primerjali z dejanskimi tekmami.

Verodostojnost generiranih simulacij smo najprej ocenili s pomočjo tradicionalnih zbirnih statistik (ang. box score), ki predstavljajo števce osnovnih dogodkov na posamezni tekmi (glej tabelo A.1 v dodatku). Primerjali smo porazdelitev vrednosti zbirnih statistik iz simuliranih tekem s porazdelitvami na dejanskih tekmah. Kot mero različ-

[3] Teoretično bi morali dobiti 24600 simulacij, toda začetnih tekem v sezoni ni možno simulirati, saj za njih ne moremo določiti vrednosti atributov.

nosti med diskretnima porazdelitvama S (empirično dobljena iz generiranih simulacij) in E (empirično dobljena iz dejanskih tekem), smo uporabili Kullback-Leiblerjevo (KL) divergenco[4]:

$$D_{KL}(E||S) = \sum_i (E(i) log \frac{E(i)}{S(i)}, \tag{5.3}$$

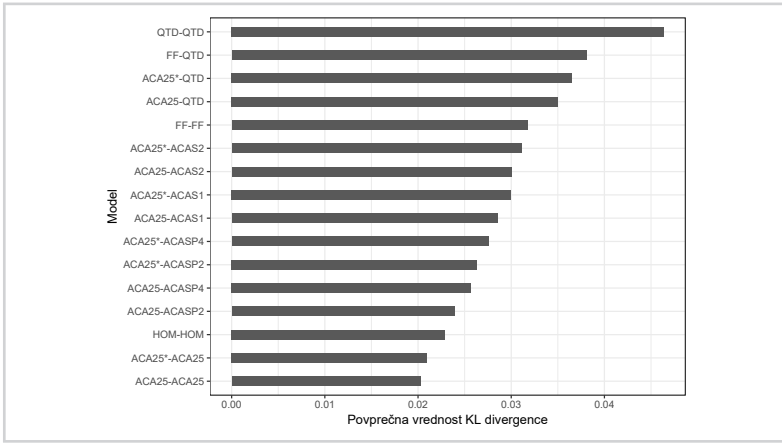
ki jo lahko interpretiramo kot količino izgubljene informacije, ko za aproksimacijo porazdelitve E uporabimo porazdelitev S.

V tabelah A.2 in A.3 v dodatku so zbrane vrednosti KL divergence za statistike domače in gostujoče ekipe. Na sliki 5.5 je prikazana primerjava med različnimi modeli glede na povprečno vrednost KL divergence z upoštevanjem vseh statistik. Podobno velja za sliko 5.6, s to razliko, da smo dejanske vrednosti KL divergence za posamezno statistiko zamenjali z rangom glede na vrstni red v primerjavi med različnimi modeli. Rezultati nakazujejo, da so pri večini statistik največja odstopanja opažena v simulacijah, generiranih z modeli, ki za napovedovanje pretoka časa uporabljajo komponento *QTD*. Najbolje se obnesejo modeli, ki uporabljajo avtomatsko generirane atributne prostore *ACASP2*, *ACASP4* in *ACA25*, medtem ko so modeli na podlagi atributnih prostorov *ACAS1* in *ACAS2* manj uspešni. Povprečne vrednosti statistik so prikazane na sliki 5.7 (za drugačen prikaz glej sliki A.3 in A.4 v dodatku). Iz slike je razvidno, da vsi modeli generirajo tekme, ki v povprečju vsebujejo preveč metov iz igre (statistiki hfga in afga), kar ima za posledico tudi nekoliko precenjeno število doseženih košev na simuliranih tekmah (glej sliko 5.8). Ostale statistike so primerljive z empiričnimi podatki.

V naslednjem eksperimentu smo generirane simulacije primerjali z empiričnimi rezultati na podlagi nekaterih karakteristik dinamike igre in gibanja rezultata, ki sta jih predstavila Gabel in Redner [51].

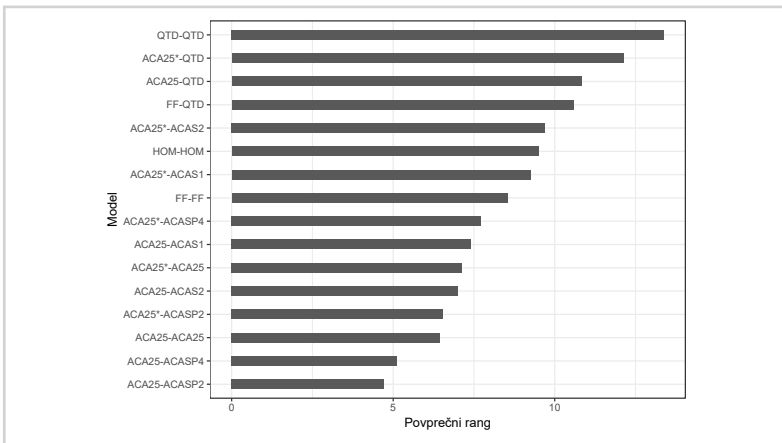
Najprej smo se osredotočili na statistične lastnosti, ki se nanašajo na posest žoge. Le-ta je definirana kot obdobje tekme, ko je žoga v popolni kontroli ene ekipe - posest se začne, ko ekipa pridobi nadzor nad žogo, in se zaključi, ko žogo prevzame njihov nasprotnik. Posest je lahko sestavljena iz večih akcij, če po zgrešenih metih iz igre sledijo uspešni skoki v napadu. Posest označimo kot uspešno, če ekipa v posesti žoge doseže točke.

[4] KL divergenca ni simetrična, vendar te lastnosti ne potrebujemo, saj je primerjava enosmerna.



Slika 5.5

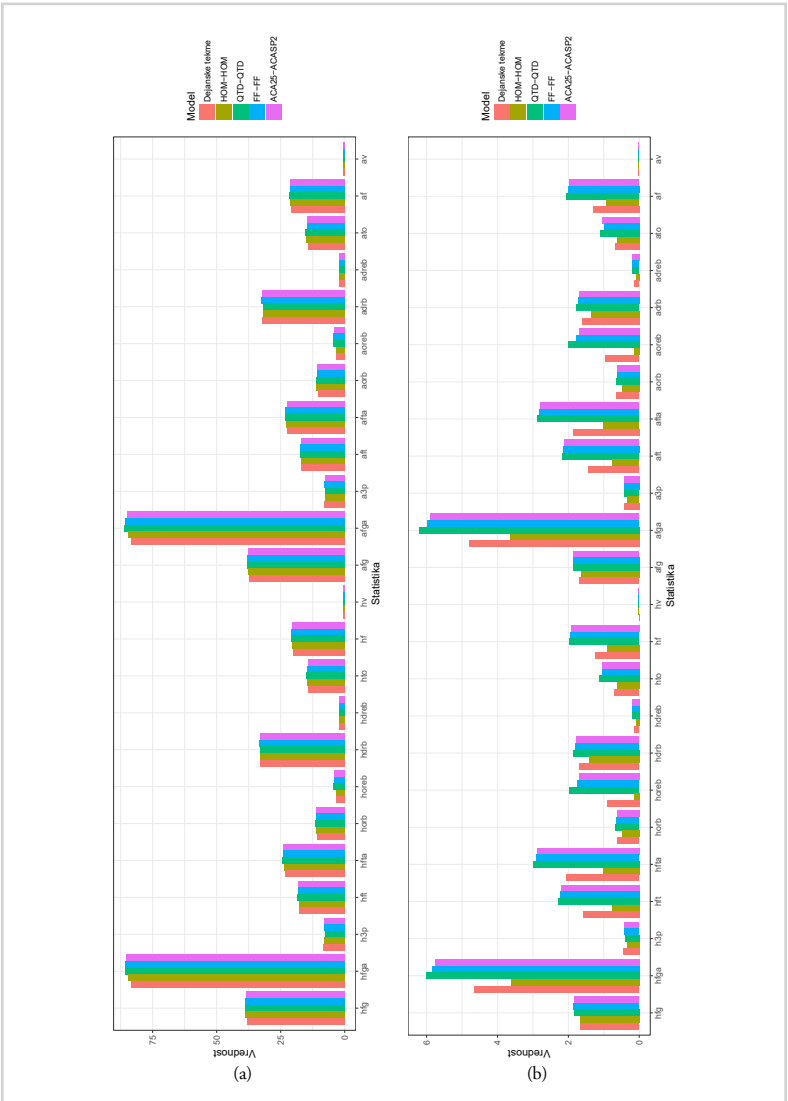
Primerjava modelov glede na povprečno vrednost KL divergence med porazdelitvami osnovnih košarkarskih statistik, ki so izmerjene na dejanskih in simuliranih tekmah (združeni sezoni NBA 2014/2015 in 2015/2016).



Slika 5.6

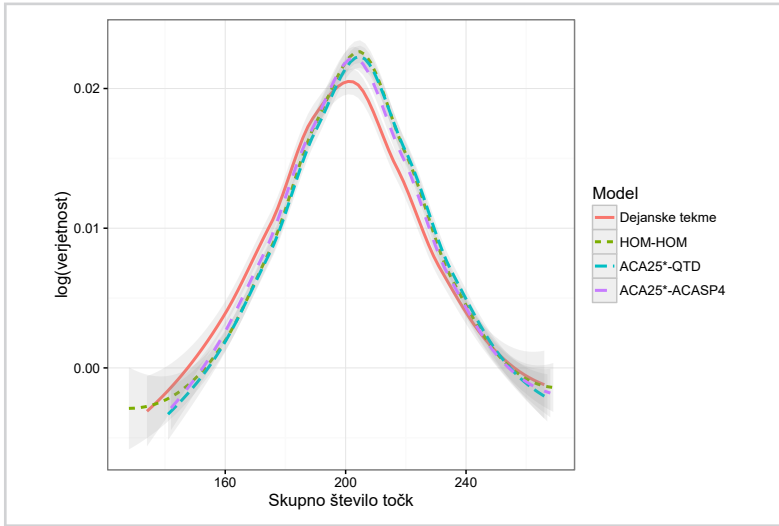
Primerjava modelov glede na povprečni rang KL divergence med porazdelitvami osnovnih košarkarskih statistik, ki so izmerjene na dejanskih in simuliranih tekmah (združeni sezoni NBA 2014/2015 in 2015/2016).

Slika 5.9 kaže porazdelitev dolžine posesti v sekundah igralnega časa. Velikih razlik med modeli ni - vse simulacije vsebujejo premalo srednje dolgih posesti (med 10 in 30 sekund), medtem ko je tistih krajših oziroma daljših preveč. Na sliki 5.10 je prikazana porazdelitev doseženih točk na uspešno posest. Rezultat sugerira, da vsi modeli generirajo premalo situacij, ko izkupiček uspešne posesti znaša tri točke, in preveč tistih, ko uspešna posest ekipi prinese le eno točko.



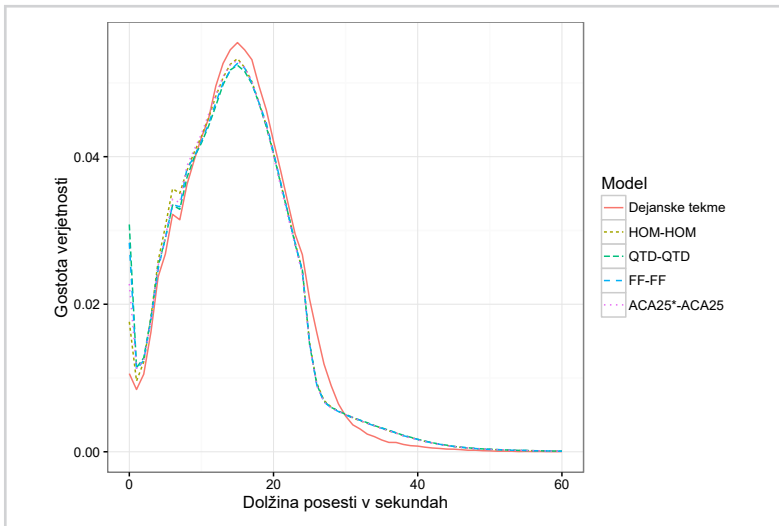
Slika 5.7

Primerjava modelov glede na povprečne vrednosti osnovnih košarkarskih statistik, ki so izmerjene na dejanskih in simuliranih tekmah (sezoni NBA 2014/2015 in 2015/2016) z upoštevanjem a) celotnega igralnega časa in b) zadnjih 30 sekund posameznih četrtin.



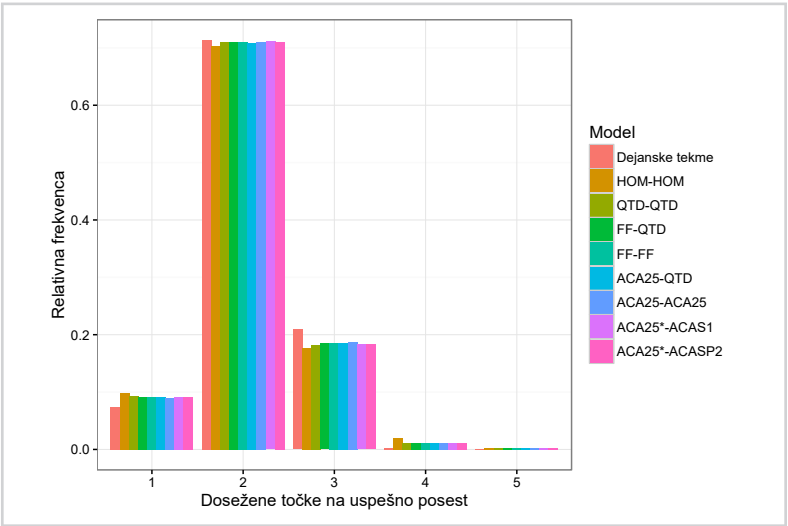
Slika 5.8

Porazdelitev skupnega števila doseženih košev na eni tekmi. Rezultati so zglejeni po metodi LOESS (faktor glajenja 0.75).



Slika 5.9

Porazdelitev dolžine posesti.

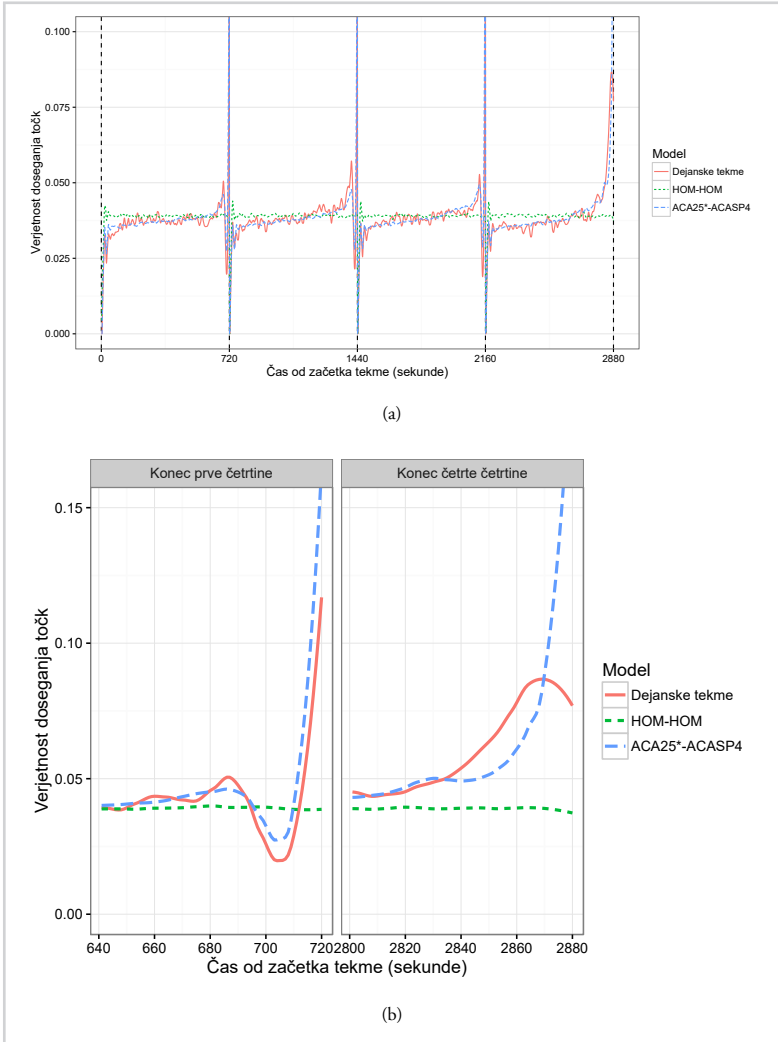


Slika 5.10

Porazdelitev doseženih točk na uspešno posest. Verjetnosti za 6 in več točk so zanemarljive in zato niso prikazane.

Dinamika doseganja točk kot funkcija igralnega časa je prikazana na sliki 5.11(a). Empirični podatki kažejo, da se v vseh četrtinah ta dinamika razvija na podoben način. Trend je konstanten ali rahlo naraščujoč tekom večjega dela igralnega časa, odstopanja so razvidna samo na začetku in na koncu vsake četrtine. Oster dvig trenda proti koncu četrtine lahko pripišemo želji ekip, da zaključijo napad pred iztekom igralnega časa. V prvih treh četrtinah je opazen tudi nenaden padec trenda, ki nastopi približno 20 sekund pred iztekom časa. Do tega efekta pride, ker ekipe namerno zavlačujejo z zaključnim metom in tako onemogočijo nasprotniku izvedbo še enega napada. Tega efekta na koncu zadnje četrtine ni, zaradi pripravljenosti ekipe v zaostanku na izvajanje hitrejših in riskantnejših akcij ter zaradi povečanega števila prostih metov, ki so posledica hitrih osebnih napak. Iz slike 5.11(a) je razvidno, da model *HOM-HOM* ni zajel opisane dinamike v empiričnih podatkih, ter da producira manj realistične simulacije s konstanim trendom doseganja točk. Po drugi strani je model *ACA25*-ACASP4* uspešno zajel spremembe trenda, vključno s padci pred koncem prvih treh četrtin. Slika 5.11(b), ki prikazuje dogajanje ob koncu prve in četrte četrtine, razkriva, da model ni pravilno zajel trenda ob koncu rednega dela tekme. Karakteristike ostalih modelov (naučenih z avtomatskimi in ekspertnimi atributi) so zelo podobne modelu *ACA25**-

ACASP4 in so zaradi večje preglednosti slike izpuščeni.

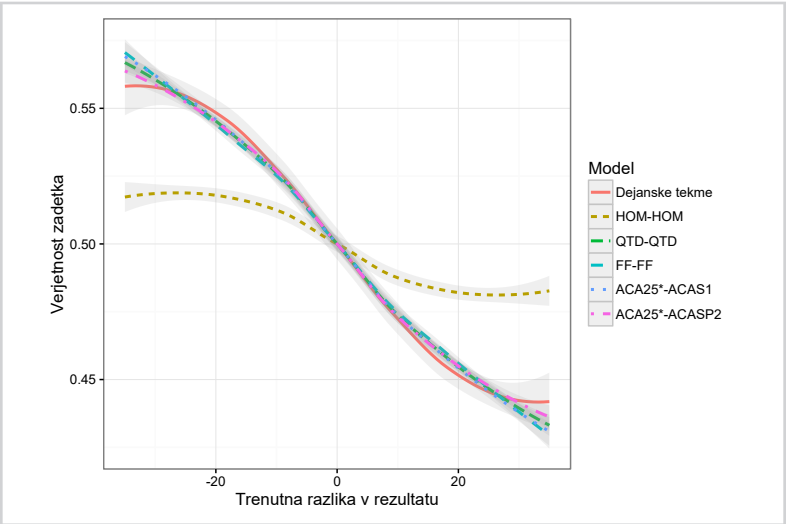


Slika 5.11

a) Dinamika doseganja točk kot funkcija igralnega časa. Rezultati so zglejeni po metodi LOESS s faktorjem glajenja 0.05. b) Povečan prikaz dinamike doseganja točk ob koncu prve in koncu četrte četrtine.

Zanimiva lastnost košarkarske igre je, da verjetnost zadetka vodilne ekipe pada z

večanjem trenutne razlike v rezultatu. Velja tudi obratno, da se verjetnost zadetka povečuje z večanjem rezultatskega zaostanka. Posledica tega efekta je dejstvo, da gibanje razlike v rezultatu kaže tendenco proti neodločenemu izidu. Slika 5.12 prikazuje, da vsi modeli izkazujejo to lastnost, čeprav je njen učinek pri modelu *HOM-HOM* nekoliko manj izrazit.

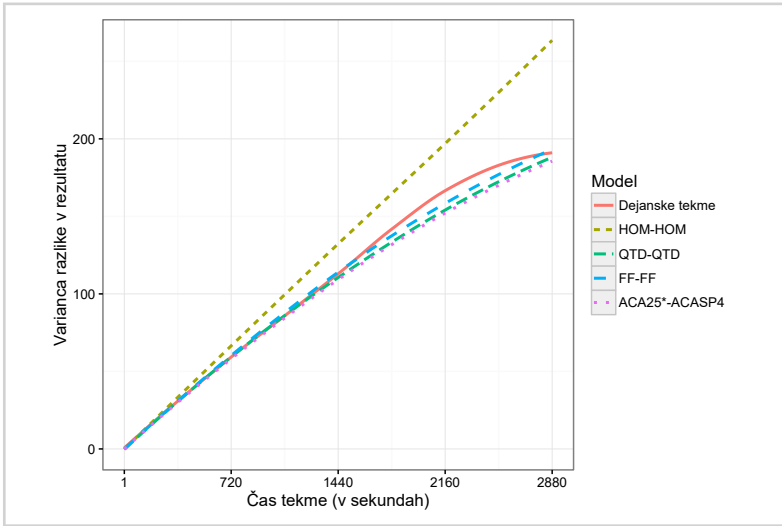


Slika 5.12

Verjetnost zadetka v odvisnosti od trenutnega rezultata. Rezultati so zglejeni po metodi LOESS (faktor glajenja 0.75).

Slika 5.13 prikazuje varianco razlike v rezultatu v odvisnosti od igralnega časa. Iz slike je razvidno, da model *HOM-HOM* generira najbolj nestabilne simulacije, medtem ko ostali modeli relativno uspešno zajamejo vzorec iz empiričnih podatkov (v drugem polčasu nekoliko podcenijo varianco razlike v rezultatu, kar je najbolj izrazito v prvi polovici zadnje četrtine).

Preverili smo tudi, v kolikšni meri simulacije povzemajo dinamiko izmenjave vodilnega moštva med potekom tekme. Vsi modeli generirajo simulacije z realističnim številom zamenjav vodilnega moštva (glej sliko 5.14). Model *HOM-HOM* generira preveč simulacij z malim številom rezultatskih preobratov (do 5), medtem ko ostali modeli generirajo nekoliko premalo takšnih tekem. Porazdelitev časa, ki ga moštvo preživi v vodstvu, je prikazana na sliki 5.15. Vse generirane simulacije so skladne z empiričnimi podatki, ki sugerirajo, da je za ekipo bolj verjetno, da bo vodilna večji



Slika 5.13

Varianca razlike v rezultatu glede na igralni čas.

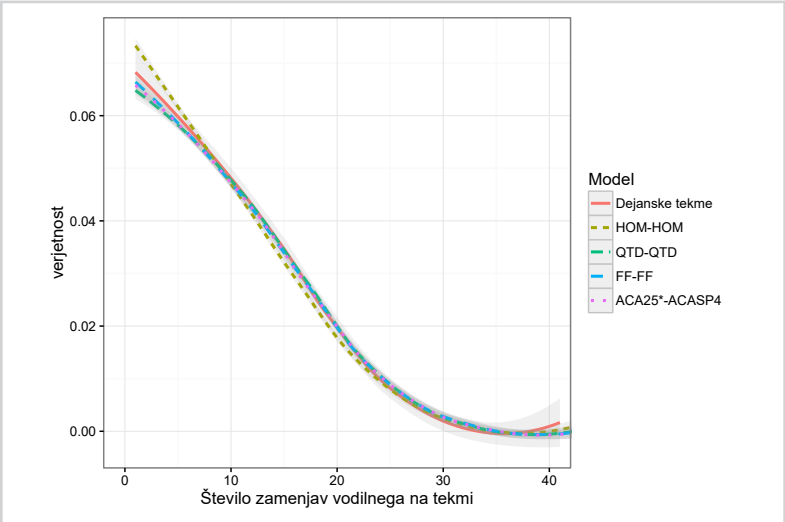
del tekme, kot da si bo čas v vodstvu enakomerno razdelila s svojim nasprotnikom. Končno, slika 5.16 prikazuje porazdelitev največjega vodstva na tekmi.

Numerična primerjava podobnosti nekaterih porazdelitev, dobljenih iz generiranih simulacij in dejanskih tekem, je predstavljena v tabeli A.4 v dodatku. Povprečen rang modelov na podlagi primerjave z empiričnimi podatki je prikazan na sliki 5.17.

5.3.5 Napovedovanje zmagovalca

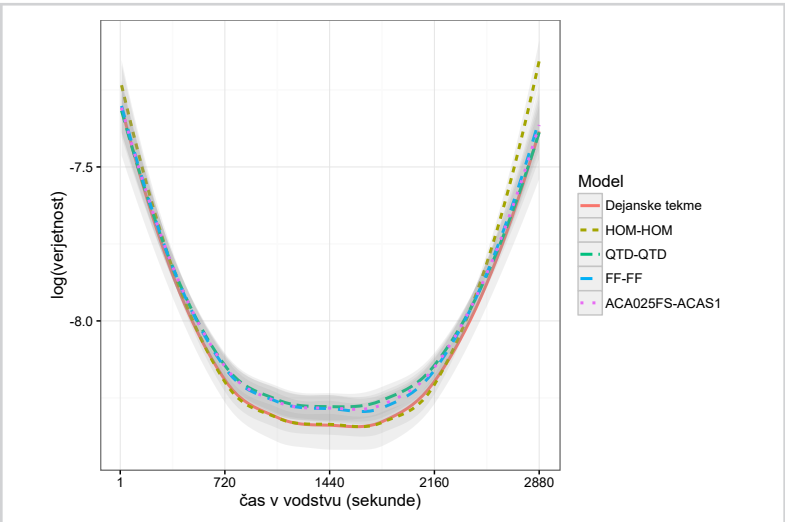
Kvaliteto napovedovanja zmagovalca lahko uporabimo kot še eno možno oceno verodostojnosti generiranih simulacij. Od dobrega simulatorja pričakujemo, da bodo izidi generiranih tekem v povprečju skladni z rezultati dejanskih tekem.

Posamezen par modelov M_{Evt}-M_{Dur} smo ovrednotili glede na kvaliteto napovedovanja zmagovalca tekme s pomočjo naslednjega postopka. Za vsako tekmo iz sezon 2014/2015 in 2015/2016 smo generirali po 1000 simulacij. Delež simuliranih tekem, ki jih je zmagalo domače moštvo, smo uporabili kot verjetnostno napoved za zmago domačina. Nato smo uporabili Brierjevo mero za oceno kvalitete napovedi kot povprečno kvadratno razliko med napovedanimi verjetnostmi zmag domačinov in dejanskimi izidi tekem. Kvaliteto simulacij smo ocenili tudi s srednjo kvadratno napako



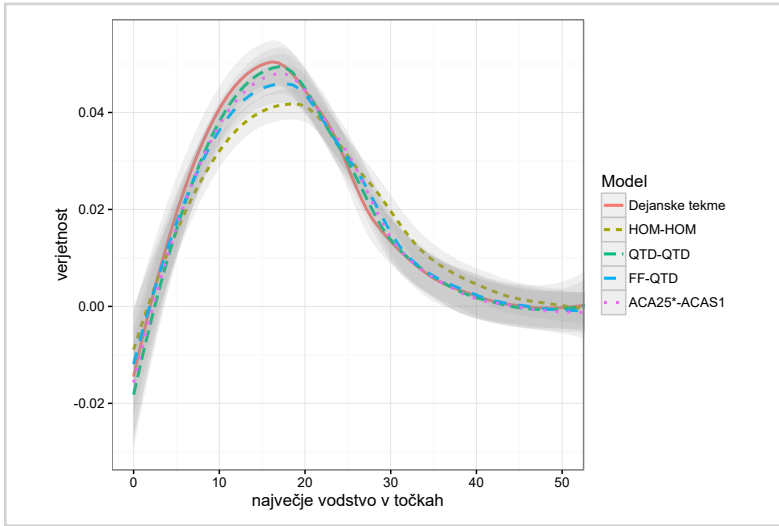
Slika 5.14

Porazdelitev števila zamenjav vodilnega moštva na tekmi. Rezultati so zglejeni po metodi LOESS (faktor glajenja 0.75).



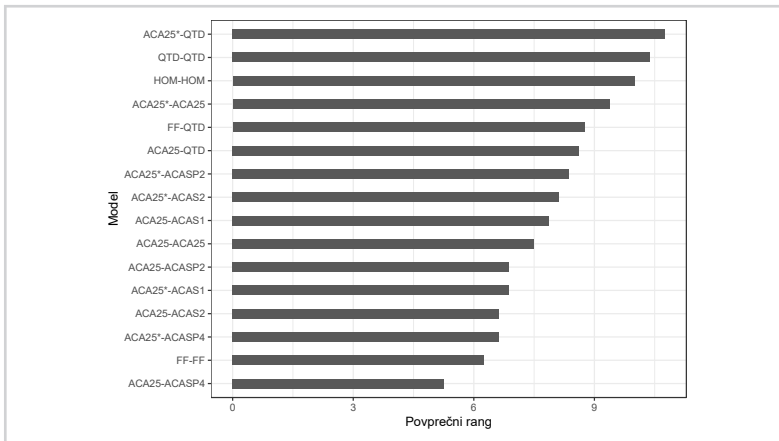
Slika 5.15

Verjetnost, da bo naključno izbrana ekipa vodila določeno število sekund med tekmo. Rezultati so zglejeni po metodi LOESS (faktor glajenja 0.75).



Slika 5.16

Porazdelitev največjega vodstva na tekmi (v točkah). Rezultati so zglejeni po metodi LOESS (faktor glajenja 0.75).



Slika 5.17

Primerjava modelov glede na povprečen rang KL divergence med porazdelitvami košarkarskih statistik za opis dinamike doseganja točk, ki so izmerjene na dejanskih in simuliranih tekmah (sezoni NBA 2014/2015 in 2015/2016).

med povprečno razliko v rezultatu simuliranih tekem in razliko v rezultatu dejanskih tekem (oboje iz perspektive domače ekipe).

Podobno kot pri prejšnjih eksperimentih smo pri generiranju simulacij tekem iz ene

sezone uporabili modele, naučene na prejšnjih dveh sezonah. Za začetne tekme v sezoni ne moremo določiti vrednosti atributov, zato jih ne moremo simulirati. Iz testa smo izločili dejanske tekme, ki so po rednem delu končane brez zmagovalca. Tako smo kvaliteto napovedovanja zmagovalca za sezono 2014/2015 določili na podlagi 1117 parov, medtem ko smo za sezono 2015/2016 uporabili 1113 parov (od teoretičnih 1230 tekem, odigranih v posameznih sezonah). Dobljeni rezultati so zbrani v tabeli 5.8.

Za primerjavo, model, ki vedno napove zmago domačina v skladu z deležem domačih zmag v učnih podatkih, za sezono 2014/2015 doseže klasifikacijsko točnost 0.58, povprečno Brierjevo mero 0.244 (± 0.0001) in MSE razlike v rezultatu 191.2 (± 8.1). Za sezono 2015/2016 pa so rezultati tega modela: klasifikacijska točnost 0.59, povprečna Brierjeva mera 0.242 (± 0.0001) in MSE razlike v rezultatu 186.1 (± 8.2).

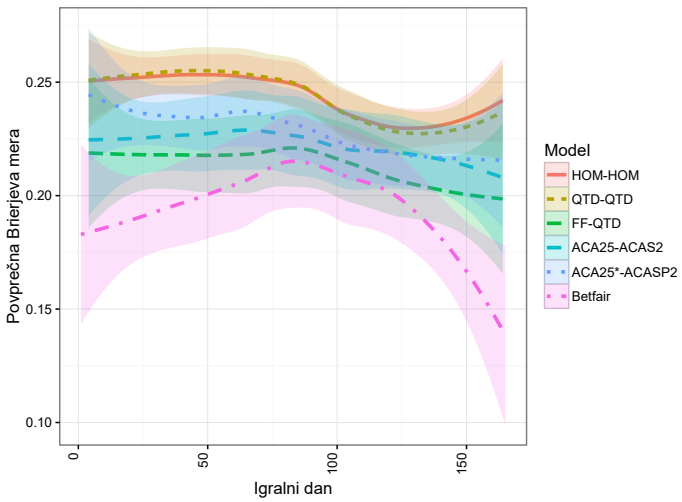
Najboljši javni vir napovedi športnih izidov so kvote stavnic. Stavna borza Betfair je za sezono 2014/2015 dosegla klasifikacijsko točnost 0.71 in povprečno Brierjevo mero 0.196 (± 0.0051).

Na slikah 5.18(a) in 5.18(b) je prikazano gibanje povprečne Brierjeve mere po igralnih dneh rednega dela sezon NBA 2014/2015 in 2015/2016. Model *FF-QTD*, ki je naučen na ekspertnih atributih, je najboljši napovedovalec tekom celotne sezone. Modela *HOM-HOM* in *QTD-QTD* sta pričakovano najslabša, saj ne upoštevata dejanskih karakteristik ekip in vedno napovedujeta povprečen izid. Modela, naučena na avtomatsko zgrajenih atributih, sta po uspešnosti nekje vmes. Na sliki 5.18(a) je prikazana tudi točnost napovedovanja stavne borze Betfair. Slika sugerira, da avtomatski modeli v primerjavi s stavnico največ izgubijo na začetku in na koncu sezone. Manj točne napovedi na začetku sezone lahko pripišemo nestabilnim atributom za opis zmogljivosti moštev, medtem ko so slabše napovedi proti koncu sezone posledica neupoštevanja faktorja motivacije. Ekipe z zagotovljeno uvrstitvijo v končnico sezone (play-off) in tiste, ki so brez možnosti za uvrstitev, ne igrajo na vso moč. Tega dejstva modeli ne upoštevajo in posledično precenjujejo možnosti nemotiviranih ekip.

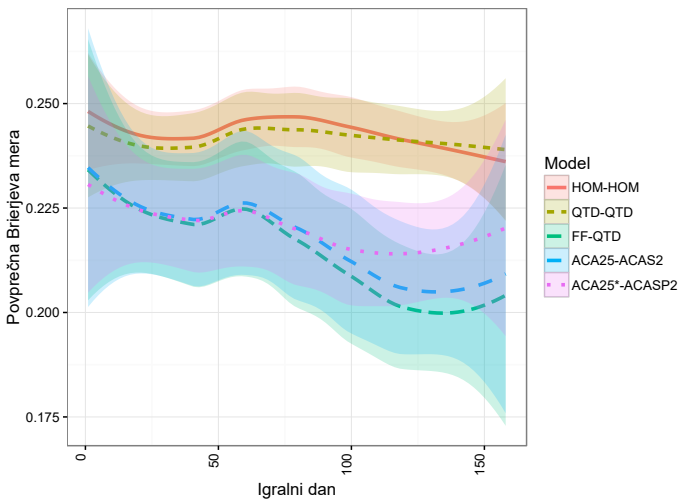
Tabela 5.8

Evalvacija modelov na podlagi napovedovanja zmagovalca, izmerjeno na tekmah rednega dela lige NBA v sezonah 2014/2015 in 2015/2016. Klasifikacijska točnost in povprečna Brierjeva mera se nanašata na napovedano verjetnost zmage domačega moštva. MSE (srednja kvadratna napaka) se nanaša na napovedano razliko v rezultatu iz perspektive domačega moštva. Vrednosti v oklepajih predstavljajo standardno napako.

	2014/15			2015/16		
Model	Točnost	Brier score	MSE	Točnost	Brier score	MSE
ACA25-ACASP2*	0.64	0.228 (± 0.0036)	178 (± 7.6)	0.65	0.220 (± 0.0035)	163 (± 7.5)
ACA25-ACAS2*	0.63	0.228 (± 0.0036)	178 (± 7.6)	0.65	0.220 (± 0.0036)	163 (± 7.5)
ACA25-QTD*	0.64	0.227 (± 0.0036)	178 (± 7.6)	0.65	0.219 (± 0.0035)	163 (± 7.4)
ACA25-ACASP2	0.62	0.224 (± 0.0043)	175 (± 7.6)	0.65	0.218 (± 0.0044)	161 (± 7.1)
ACA25-ACAS2	0.63	0.223 (± 0.0043)	175 (± 7.7)	0.65	0.217 (± 0.0044)	162 (± 7.2)
ACA25-ACA25	0.63	0.223 (± 0.0043)	174 (± 7.7)	0.64	0.218 (± 0.0044)	162 (± 7.1)
ACA25-QTD	0.63	0.225 (± 0.0043)	175 (± 7.6)	0.66	0.217 (± 0.0045)	161 (± 7.1)
FF-FF	0.65	0.215 (± 0.0042)	168 (± 7.5)	0.65	0.214 (± 0.0043)	157 (± 7.0)
FF-QTD	0.66	0.215 (± 0.0042)	168 (± 7.6)	0.65	0.213 (± 0.0042)	156 (± 7.0)
QTD-QTD	0.58	0.244 (± 0.0026)	191 (± 8.1)	0.59	0.241 (± 0.0022)	186 (± 8.2)
HOM-HOM	0.58	0.244 (± 0.0022)	192 (± 8.1)	0.59	0.243 (± 0.0019)	186 (± 8.2)
		N = 1117			N = 1113	



(a)

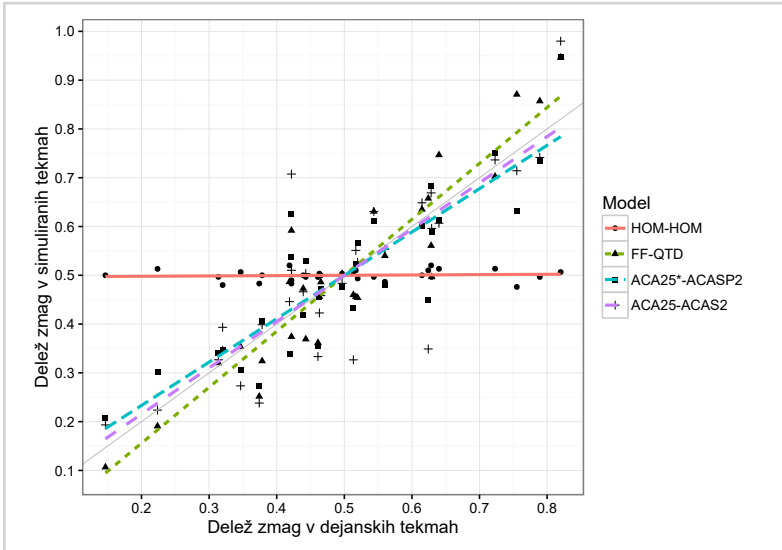


(b)

Slika 5.18

Povprečna vrednost Brierjeve mere pri napovedovanju verjetnosti zmage domačega moštva na tekmah rednega dela lige NBA v sezonah a) 2014/2015 in b) 2015/2016. Rezultati so zglajeni po metodi LOESS (faktor glajenja 0.75)..

Za vsa moštva v ligi NBA smo primerjali napovedan in dejanski delež zmag proti povprečnemu nasprotniku (uporabili smo tekme obeh testnih sezon). Rezultat na sliki 5.19 kaže, da so modeli, naučeni na ekspertnih in avtomatsko zgrajenih atributih, dobri napovedovalci deleža zmag. Model *FF-QTD* v povprečju nekoliko precenjuje močnejše in podcenjuje slabše ekipe, medtem ko za modela *ACA25-ACAS2* in *ACA25*-ACASP2-QTD* velja obratno.



Slika 5.19

Razmerje med dejanskim in napovedanim deležem zmag za tekme rednega dela lige NBA v sezonah 2014/2015 in 2015/2016, ki so se končale v rednem delu.

6

Zaključek in nadaljnje delo

6.1 Zaključek

V delu smo se ukvarjali z modeliranjem razvoja dogodkov na športni tekmi. Izhajali smo iz markovskega modela, pri čemer smo prostor stanj razširili z opisom trenutnega konteksta tekme. Na ta način smo v opis stanja vključili tudi del zgodovine razvoja procesa (tekme) in s tem omilili markovsko lastnost modela, ki določa, da je verjetnost prehoda v naslednje stanje odvisno samo od trenutnega stanja.

Opisali smo kodiranje prostora stanj v obliki vektorja, sestavljenega iz spremenljivk treh tipov: (1) slučajne spremenljivke, ki neposredno vplivajo na razvoj dogodkov na tekmi, (2) spremenljivke, ki vplivajo na razvoj dogodkov na tekmi, vendar niso slučajne (so posledica prejšnjih dogodkov na tekmi) in (3) parametri, ki vplivajo na potek tekme, vendar se njihova vrednost med samo tekmo ne spreminja.

Predstavili smo postopek za modeliranje matrike prehoda markovskega modela v odvisnosti od komponent opisa trenutnega stanja. Osnovna ideja temelji na uporabi kaskade zaporedno vezanih modelov. Prvi model v kaskadi na podlagi opisa trenutnega stanja oceni marginalno pogojno porazdelitev prve slučajne spremenljivke iz opisa naslednjega stanja. Vsak naslednji model ocenjuje porazdelitev naslednje slučajne spremenljivke v odvisnosti od trenutnega stanja in pogojeno z izbranimi vzorci iz napovedi predhodnih modelov v kaskadi. Ko zberemo vzorce vseh modelov v kaskadi, lahko rekonstruiramo celoten opis naslednjega stanja.

Modele iz kaskade lahko naučimo vsakega posebej, saj so medsebojno neodvisni. Zaradi diskretne narave pravil športnih iger smo predlagali drevesne strukture kot primerne modele za modeliranje razvoja športnih dogodkov. Opisali smo postopek, ki na podlagi naučenih modelov generira simulacije tekem med specifičnima ekipama iz poljubno definirane izhodiščne situacije.

Za učenje modelov potrebujemo atributni prostor, ki predstavlja komponente opisa prostora stanj. Običajno se v atributni prostor vključijo opisne statistike, ki določajo zmogljivosti nasprotnih ekip in so največkrat podane s strani domenskih ekspertov. Predstavili smo postopek za avtomatsko generiranje atributnega prostora, ki ne potrebuje domenskega predznanja. Osnovna ideja temelji na identifikaciji množic podobnih dogodkov, pri čemer se podobnost nanaša na porazdelitve, ki opisujejo predhodne oziroma naslednje dogodke v opazovanih sekvencah razvoja dejanskih tekem. Vsaka množica podobnih dogodkov opredeljuje nek abstraktni koncept. Opisali smo postopek, ki najprej s pomočjo hierarhičnega razvrščanja identificira abstraktne koncepte,

nato pa generira atributni prostor v obliki razmerij med številom vstopov in izstopov iz teh konceptov. Postopek v zadnjem koraku izbere ustrezno podmnožico generiranih atributov na podlagi kriterijev splošnosti, maksimizacije entropije in komplementarnosti. Na košarkarski domeni smo pokazali, da je atributni prostor, dobljen s predlaganim postopkom, po strukturi zelo podoben ekspertnim statistikam na podlagi štirih faktorjev.

Eksperimentalna evalvacija na košarkarski domeni (izvedena na sezonah lige NBA 2014/2015 in 2015/2016) je pokazala, da so modeli, dobljeni z učenjem na podlagi avtomatsko generiranih atributov po kvaliteti napovedovanja naslednjega dogodka in časa med dogodki primerljivi z modeli, naučenimi z ekspertnimi atributi.

Statistična analiza dobljenih simulacij je pokazala, da so modeli uspešno zajeli dinamiko razvoja košarkarske tekme in da se nekateri karakteristični vzorci dobro ujemajo z empiričnimi podatki. Ugotovili smo, da je razvoj košarkarske tekme večinoma homogen proces z izjemo začetka in konca vsake četrtine. Kljub temu modeliranje teh nehomogenih delov razvoja tekme bistveno izboljša verodostojnost generiranih simulacij.

Modele smo primerjali tudi po kvaliteti napovedovanja zmagovalca in ugotovili, da so modeli, naučeni na ekspertnih atributih, nekoliko boljši napovedovalci verjetnosti zmage domačina. Eksperimentalna evalvacija je pokazala, da so stavne borze najbolj točni javni napovedovalci zmagovalca tekem lige NBA, kar je v skladu z dosedanjimi raziskavami. Ugotovili smo tudi, da modeli slabše napovedujejo predvsem tekme v začetnem in končnem delu rednega dela sezone. Slabše napovedi v začetnem delu sezone smo pripisali nestabilnim in nezanesljivim vrednostim atributov za opis zmogljivosti ekip, medtem ko smo nenatančne napovedi na koncu sezone razložili z neupoštevanjem pomembnosti tekem v širšem kontekstu tekmovanja (uvrstitev v končnico sezone).

Predlagana metodologija je splošna in jo lahko uporabimo pri modeliranju vseh športov, ki jih je možno dobro opisati z markovskim prehodom v prostoru stanj. Način kodiranja prostora stanj omogoča modeliranje vseh pomembnih vidikov razvoja tekme (na primer: kaj se bo zgodilo v naslednjem koraku, kje se bo zgodilo, kdaj se bo zgodilo itd.) in vključitev relevantnih faktorjev, ki krmilijo potek dogodkov na igrišču. Prav tako je splošna tudi metoda za avtomatsko generiranje atributnega prostora, ki jo lahko uporabimo za modeliranje kateregakoli vidika razvoja tekme, tudi pri športih, za katere niso znani dobri ekspertni atributi za opis zmogljivosti ekip. Ocenjujemo, da je predlagana metodologija ustrezna za modeliranje najpopularnejših ekipnih športov,

kot so nogomet, ameriški nogomet, košarka, rokomet, hokej, vaterpolo, odbojka itd. Zaradi svoje splošnosti bi metodo za gradnjo atributnega prostora lahko uporabili tudi pri drugih (ne športnih) domenah, ki jih modeliramo s pomočjo stohastičnih procesov (npr. finančni in ekonomski podatki v obliki časovnih vrst).

Predstavljena metodologija ima nekaj omejitev. Razvita je za modeliranje play-by-play zapisov o razvoju športnih tekem, ki tipično podpirajo nekaj deset različnih tipov dogodkov. Če želimo generirati podrobnejše simulacije, je potrebno razširiti nabor osnovnih dogodkov, kar potencijalno oteži identifikacijo pomembnih konceptov in s tem tudi atributov za modeliranje poteka dogodkov. Postopek za grupiranje osnovnih dogodkov glede na podobnost temelji na lokalni soseščini. Posledično, v primeru prevelikega števila osnovnih dogodkov, ne bomo identificirali globalnih konceptov, ki dejansko vplivajo na dogajanje na tekmi, ampak bomo dobili množico lokalnih in največkat irelevantnih značilk.

Iz lokalnosti soseščine izvira tudi težava pri generiranju atributnega prostora za modeliranje zveznih komponent prostora stanj (oziroma diskretnih komponent z velikim številom različnih vrednosti, kot je na primer pozicija naslednjega dogodka na nogometnem igrišču, izražena v metrih). V tovrstnih situacijah si lahko pomagamo z bolj grobo diskretizacijo vrednosti ciljne spremenljivke, ki bo zgostila lokalno soseščino in mogoče izboljšala dobljene rezultate.

Omejitev opisane metode za generiranje atributnega prostora izvira iz uporabe trdega gručenja, ki ima za posledico, da je vsak osnovni dogodek del le enega abstraktnega koncepta. Ta omejitev lahko povzroči, da metoda izpusti potencialno uporabne koncepte oziroma informativne atribute.

6.2 Nadaljnje delo

Ugotovitve iz razdelka 6.1 predstavljajo iztočnice za nadaljnje delo. Trdo gručenje lahko zamenjamo z bolj splošnimi pristopi (na primer mehko ali nedisjunktno gručenje), ki dovoljujejo pripadnost posameznih elementov večim skupinam hkrati. Na ta način bi metoda identifikacije višjenivojskih konceptov postala bolj fleksibilna in bi omogočala identifikacijo konceptov, ki jih trenutno ni možno detektirati.

Rezultati empirične evalvacije nakazujejo, da bi z vključitvijo atributov za opis širšega tekmovalnega konteksta modeliranih tekem dobili bolj verodostojne simulacije in natančnejše napovedi zmagovalca. Tukaj mislimo predvsem na oceno pomembnosti tekme, ki lahko vpliva na motivacijo ekip. Ostali atributi za opis širšega konteksta

tekme so izčrpanost igralcev zaradi težkega razporeda in pogostih tekem, nedavne poškodbe ali kazni ključnih igralcev, ki lahko bistveno vplivajo na zmogljivosti ekipe in podobno. Težava pri tovrstnih atributih je v tem, da jih je težko avtomatsko določiti. Rešitev je lahko v razširitvi atributnega prostora s kvotami stavnic, ki naj bi upoštevale vse javno dostopne vire informacij, relevantne za končen izid tekme. Druga možnost za črpanje dodatne informacije o širšem tekmovalnem kontektu je uporaba jezikovnih tehnologij za analizo objav na družbenih omrežjih in portalih s športnimi novicami.

Dodatno izboljšavo simulatorja bi lahko dosegli, če bi namesto učenja iz točkovnih vrednosti modele gradili na podlagi verjetnostnih porazdelitev ocen zmogljivosti ekip. Dimitriev in Štrumbelj [62] sta pokazala, da ta pristop bistveno omili negativne posledice začetne negotovosti pri ocenjevanju zmogljivosti ekip in posledično izboljša kvaliteto napovedi.

Ključna ugotovitev eksperimentalne evalvacije predlagane metode je odločilna vloga modeliranja nehomogenih delov tekem na kvaliteto dobljenih simulacij. V tem smislu bi bilo zanimivo preiskusiti pristop z uporabo ločenih modelov za homogene in nehomogene dele tekme.

Predlagana metodologija izvede faktorizacijo pogojne porazdelitve po verižnem pravilu iz enačbe (3.3) v privzetem vrstnem redu slučajnih spremenljivk v vektorskem zapisu stanja. Če bi bili naučeni modeli za napovedovanje posameznih komponent pogojne porazdelitve idealno točni, bi bil vrstni red faktorizacije nepomemben. Ker je to nerealno pričakovati, vrstni red faktorizacije bistveno vpliva na kvaliteto modela v celoti. Iz tega razloga bi bilo koristno opisano metodologijo razširiti s postopkom za identifikacijo (sub)optimalnega vrstnega reda faktorizacije, ki bo zagotovil čim višjo kvaliteto napovedovanja posameznih komponent kaskade modelov.

Eno izmed izhodišč v disertaciji je interpretabilnost dobljenih modelov, ki bi potrdili obstoječe in po možnosti odkrili nove koncepte modeliranega športa. Če nas v prvi vrsti zanima kvaliteta napovedi in generiranih simulacij in ne sama interpretabilnost modelov, bi bilo zanimivo preizkusiti druge metode, ki so se pokazale uspešne v aplikacijah za generiranje kompleksnih vzorcev (na primer rekurenčne ali globoke nevronske mreže pri analizi besedil v naravnem jeziku).

Zanimiva alternativa pri modeliranju pogojne porazdelitve iz enačbe (3.3) bi bila uporaba metod večciljnega učenja (ang. multitask learning). Ideja temelji na dejstvu, da so posamezne spremenljivke v vektorskem zapisu stanja med seboj odvisne in je pričakovati, da so interni koncepti, uporabljeni pri modeliranju ene spremenljivke,

koristni tudi pri modeliranju ostalih spremenljivk.

Končno, v nadaljnjem delu bomo v proces modeliranja poteka športne tekme vključili tudi časovno-prostorske podatke o gibanju igralcev in žoge, ki so v zadnjem času postali javno dostopni in so deležni čedalje večje pozornosti raziskovalcev na področju športne analitike. Vključitev tovrstnih podatkov bi simuliranje športnih dogodkov še bolj približalo nivoju posameznih igralcev, kar bi nedvomno povečalo aplikativno uporabnost metodologije.

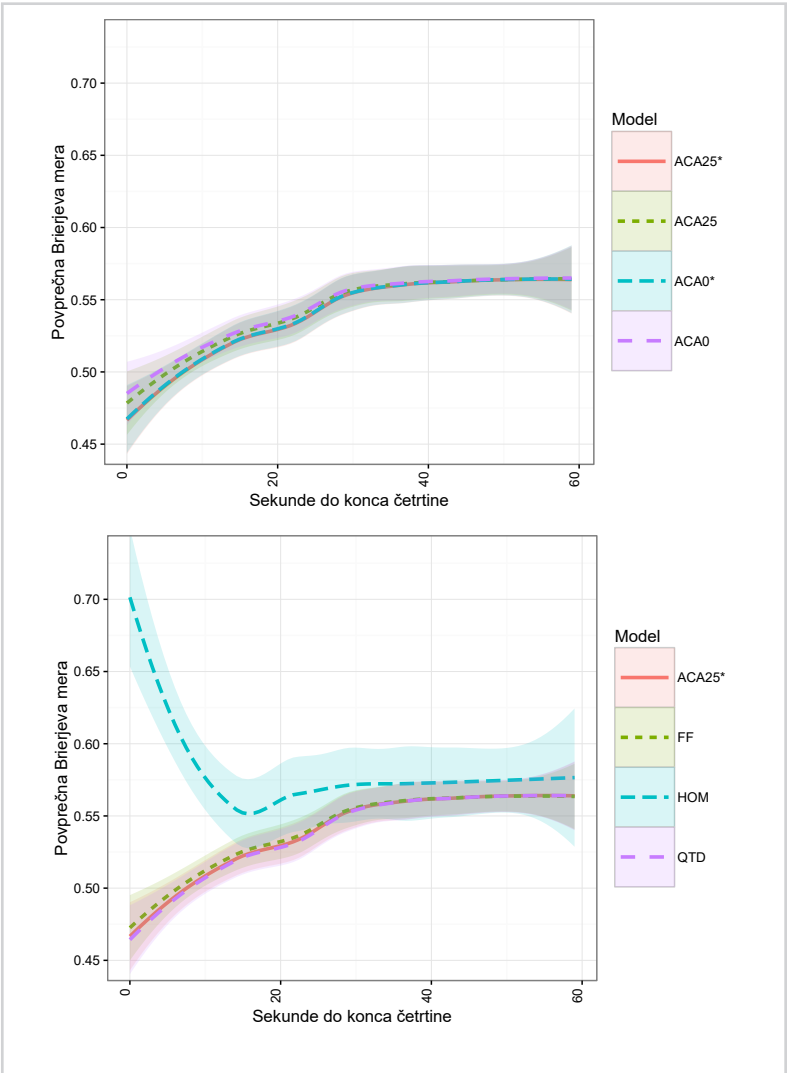
Dodatek A

V dodatku je podan opis osnovnih košarkarskih statistik, uporabljenih pri analizi posameznih vidikov verodostojnosti dobljenih simulacij. Dodatek vsebuje tudi povečane prikaze nekaterih grafov iz glavnega besedila, nekaj dodatnih slik in numerične vrednosti nekaterih primerjav med različnimi modeli.

Tabela A.1

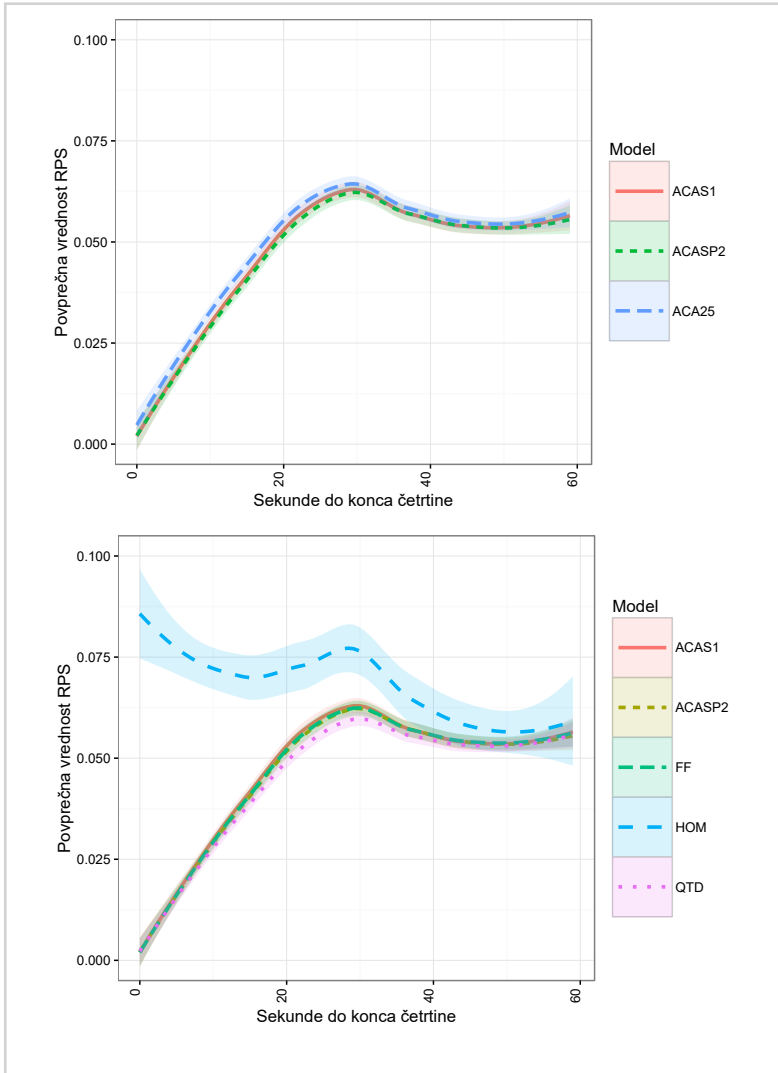
Opis osnovnih košarkarskih statistik

Oznaka statistike	Opis
afga, hfga	Število gostujočih / domačih metov iz igre.
afg, hfg	Število gostujočih / domačih uspešnih metov iz igre.
a3p, h3p	Število gostujočih / domačih uspešnih metov za tri točke.
afta, hfta	Število gostujočih / domačih prostih metov.
aft, hft	Število gostujočih / domačih uspešnih prostih metov.
adrb, hdrb	Število gostujočih / domačih skokov v obrambi.
adreb, hdreb	Število gostujočih / domačih timskih skokov v obrambi.
aorb, horb	Število gostujočih / domačih skokov v napadu.
aoreb, horeb	Število gostujočih / domačih timskih skokov v napadu.
ato, hto	Število gostujočih / domačih zapravljenih žog.
av, hv	Število gostujočih / domačih prekrškov.
af, hf	Število gostujočih / domačih osebnih napak.



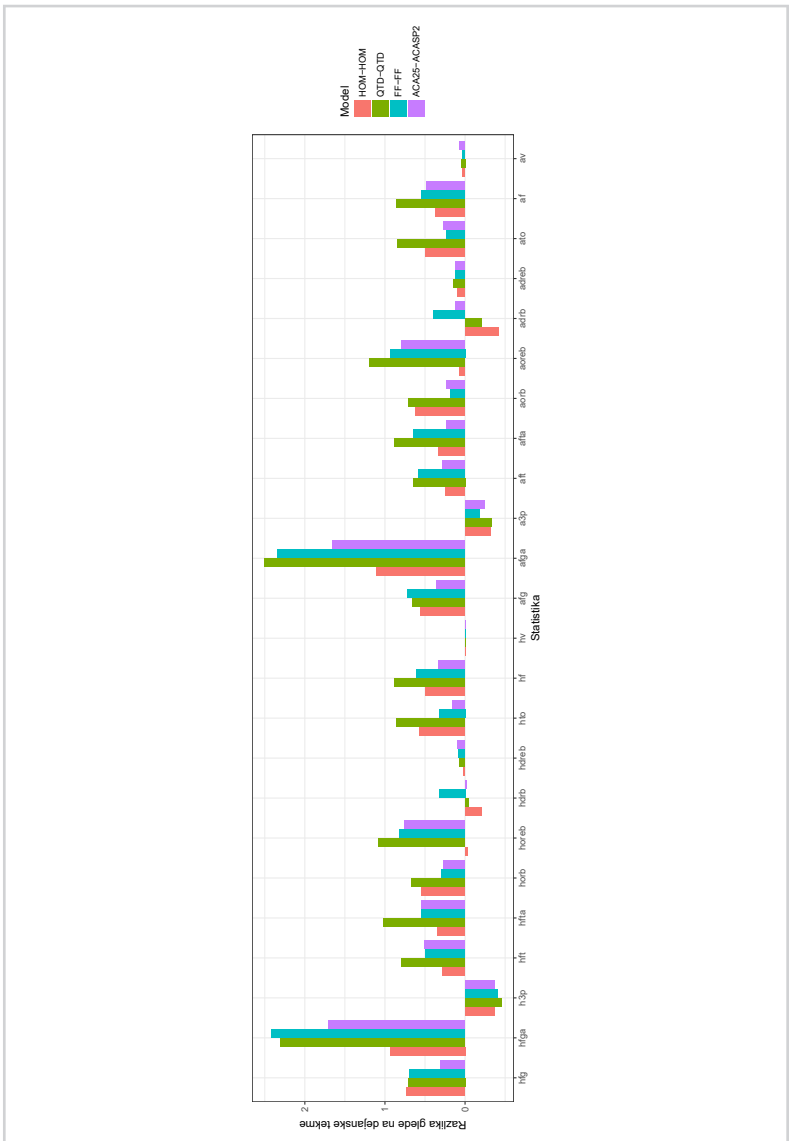
Slika A.1

Povečan prikaz dela slike 5.1, ki se nanaša na spreminjanje povprečne vrednosti Brierjeve mere v zadnji minuti četrtine.



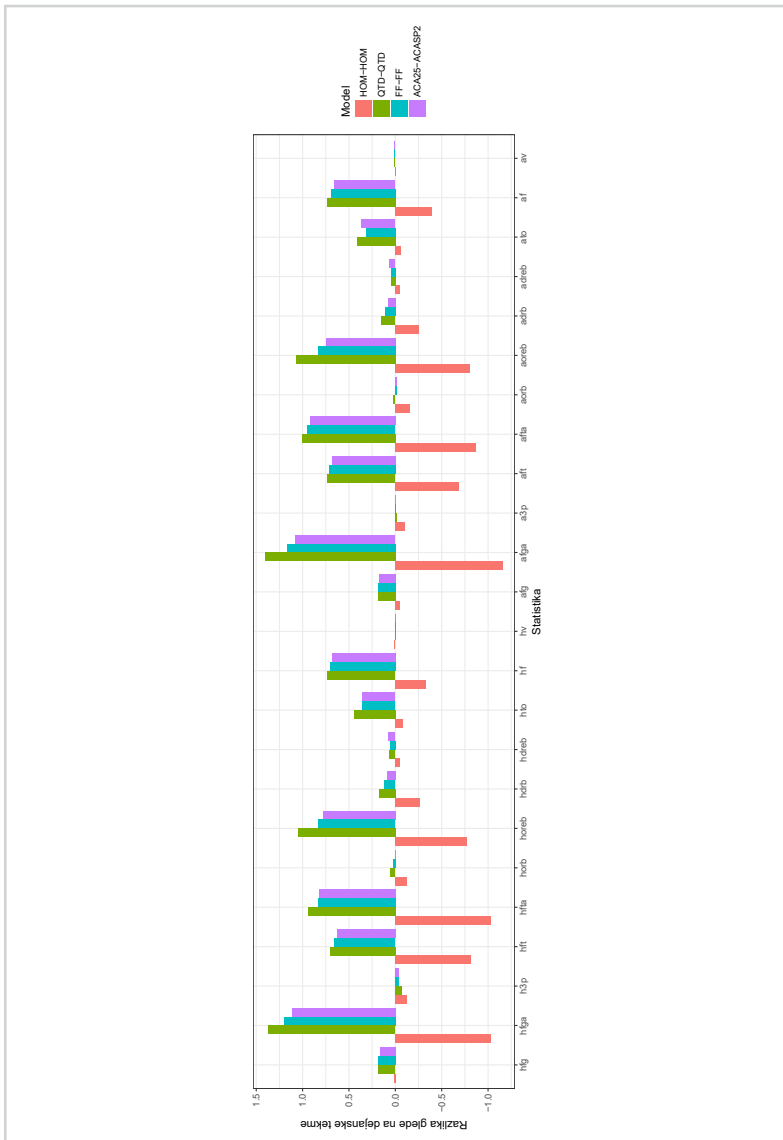
Slika A.2

Povečan prikaz dela slike 5.3, ki se nanaša na spreminjanje povprečne vrednosti ocene RPS v zadnji minuti četrtine.



Slika A.3

Primerjava modelov glede na povprečno razliko vrednosti osnovnih košarkarskih statistik med simuliranimi in dejanskimi tekmami z upoštevanjem celotnega igralnega časa. Vrednosti so izmerjene na sezonah NBA 2014/2015 in 2015/2016.



Slika A.4

Primerjava modelov glede na povprečno razliko vrednosti osnovnih košarkarskih statistik med simuliranimi in dejanskimi tekmami z upoštevanjem zadnjih 30 sekund posameznih četrtin. Vrednosti so izmerjene na sezonah NBA 2014/2015 in 2015/2016.

Tabela A.2

KL divergenca med porazdelitvami osnovnih košarkarskih statistik (domačega moštva) dejanskih in simuliranih tekem. Podatki se nanašajo na združeni sezoni NBA 2014/2015 in 2015/2016.

Model	hfg	h3p	hfga	horb	horeb	hdrb	hdreb	hft	hfta	hto	hf	hv
ACA25-ACASP4*	0.015	0.059	0.052	0.016	0.080	0.007	0.005	0.029	0.034	0.018	0.016	0.002
ACA25-ACASP2*	0.011	0.064	0.050	0.012	0.077	0.006	0.004	0.025	0.029	0.019	0.012	0.002
ACA25-ACAS2*	0.015	0.063	0.075	0.016	0.086	0.009	0.006	0.032	0.031	0.022	0.014	0.002
ACA25-ACAS1*	0.013	0.064	0.079	0.017	0.077	0.008	0.006	0.032	0.034	0.022	0.017	0.001
ACA25-ACA25*	0.023	0.061	0.059	0.018	0.003	0.009	0.003	0.024	0.027	0.017	0.014	0.001
ACA25-QTD*	0.020	0.062	0.096	0.018	0.119	0.010	0.007	0.027	0.036	0.025	0.017	0.002
ACA25-ACASP4	0.016	0.048	0.059	0.011	0.073	0.006	0.004	0.023	0.027	0.011	0.015	0.001
ACA25-ACASP2	0.017	0.046	0.057	0.012	0.072	0.007	0.005	0.026	0.024	0.010	0.011	0.002
ACA25-ACAS2	0.021	0.044	0.090	0.014	0.088	0.009	0.005	0.026	0.027	0.012	0.014	0.001
ACA25-ACAS1	0.023	0.049	0.092	0.014	0.070	0.009	0.005	0.024	0.025	0.012	0.014	0.001
ACA25-ACA25	0.024	0.040	0.071	0.015	0.004	0.010	0.004	0.019	0.027	0.011	0.015	0.001
ACA25-QTD	0.022	0.046	0.104	0.014	0.114	0.010	0.005	0.029	0.032	0.017	0.018	0.002
FF-FF	0.023	0.045	0.097	0.015	0.085	0.008	0.004	0.029	0.032	0.014	0.018	0.001
FF-QTD	0.024	0.048	0.130	0.016	0.125	0.010	0.003	0.029	0.035	0.018	0.021	0.002
QTD-QTD	0.023	0.061	0.103	0.033	0.144	0.006	0.003	0.037	0.049	0.046	0.029	0.001
HOM-HOM	0.021	0.068	0.034	0.035	0.003	0.011	0.004	0.027	0.028	0.027	0.015	0.003

Tabela A.3

KL divergenca med porazdelitvami osnovnih košarkarskih statistik (gostujočega moštva) dejanskih in simuliranih tekem. Podatki se nanašajo na združeni sezoni NBA 2014/2015 in 2015/2016.

Model	afg	a3p	afga	aorb	aoreb	adrb	adreb	aft	afta	ato	af	av
ACA25-ACASP4*	0.007	0.048	0.053	0.015	0.099	0.007	0.004	0.022	0.031	0.019	0.018	0.006
ACA25-ACASP2*	0.006	0.052	0.049	0.017	0.085	0.008	0.004	0.024	0.031	0.022	0.016	0.006
ACA25-ACAS2*	0.013	0.048	0.080	0.018	0.100	0.009	0.004	0.022	0.030	0.024	0.021	0.006
ACA25-ACAS1*	0.015	0.043	0.081	0.019	0.083	0.008	0.004	0.019	0.024	0.024	0.024	0.007
ACA25-ACA25*	0.018	0.046	0.057	0.022	0.003	0.010	0.004	0.019	0.023	0.018	0.016	0.006
ACA25-QTD*	0.013	0.047	0.097	0.019	0.133	0.012	0.005	0.022	0.031	0.029	0.023	0.007
ACA25-ACASP4	0.009	0.036	0.072	0.018	0.104	0.009	0.005	0.016	0.021	0.013	0.017	0.005
ACA25-ACASP2	0.010	0.035	0.059	0.016	0.080	0.007	0.005	0.016	0.020	0.015	0.015	0.007
ACA25-ACAS2	0.018	0.033	0.098	0.017	0.107	0.011	0.005	0.019	0.021	0.015	0.021	0.006
ACA25-ACAS1	0.018	0.032	0.093	0.017	0.084	0.009	0.005	0.020	0.025	0.017	0.021	0.006
ACA25-ACA25	0.020	0.034	0.072	0.021	0.003	0.015	0.004	0.017	0.023	0.014	0.016	0.007
ACA25-QTD	0.019	0.036	0.110	0.019	0.132	0.010	0.006	0.021	0.025	0.016	0.027	0.007
FF-FF	0.019	0.035	0.102	0.015	0.109	0.009	0.004	0.028	0.032	0.015	0.017	0.004
FF-QTD	0.021	0.034	0.113	0.017	0.151	0.010	0.004	0.028	0.035	0.015	0.018	0.005
QTD-QTD	0.019	0.049	0.117	0.041	0.176	0.010	0.007	0.035	0.043	0.046	0.027	0.008
HOM-HOM	0.012	0.054	0.048	0.042	0.003	0.014	0.004	0.026	0.024	0.027	0.012	0.006

Tabela A.4

KL divergenca med nekaterimi porazdelitvami (ocenjenimi iz generiranih simulacij) in dejanskimi podatki. A - dolžina posesti, B - čas med zaporednima košema (katerekoli ekipe), C - čas med zaporednima košema (iste ekipe), D - skupno število košev na tekmi, E - dolžina niza zaporednih košev (v točkah), F - najvišje vodstvo na tekmi, G - število izmenjav vodilnega na tekmi, H - čas v vodstvu na tekmi.

	Lastnost							
Model M_{Evt}-M_{Dur}	A	B	C	D	E	F	G	H
ACA25-ACASP4*	0.0216	0.0102	0.0064	0.0505	0.0013	0.0157	0.0091	0.3801
ACA25-ACASP2*	0.0216	0.0105	0.0069	0.0509	0.0013	0.0142	0.0107	0.3805
ACA25-ACAS2*	0.0219	0.0101	0.0072	0.0556	0.0013	0.0167	0.0105	0.3737
ACA25-ACAS1*	0.0210	0.0108	0.0076	0.0604	0.0013	0.0129	0.0092	0.3729
ACA25-ACA25*	0.0174	0.0113	0.0076	0.0612	0.0012	0.0159	0.0105	0.3721
ACA25-QTD*	0.0223	0.0109	0.0075	0.0610	0.0013	0.0154	0.0098	0.3794
ACA25-ACASP4	0.0216	0.0102	0.0063	0.0555	0.0011	0.0172	0.0080	0.3750
ACA25-ACASP2	0.0215	0.0105	0.0067	0.0483	0.0011	0.0178	0.0097	0.3765
ACA25-ACAS2	0.0219	0.0101	0.0072	0.0646	0.0011	0.0156	0.0092	0.3757
ACA25-ACAS1	0.0210	0.0108	0.0074	0.0650	0.0011	0.0153	0.0098	0.3776
ACA25-ACA25	0.0175	0.0113	0.0075	0.0626	0.0009	0.0142	0.0092	0.3764
ACA25-QTD	0.0222	0.0108	0.0074	0.0694	0.0011	0.0135	0.0099	0.3764
FF-FF	0.0206	0.0108	0.0074	0.0615	0.0012	0.0165	0.0077	0.3708
FF-QTD	0.0221	0.0110	0.0074	0.0750	0.0012	0.0156	0.0091	0.3713
QTD-QTD	0.0226	0.0111	0.0077	0.0697	0.0014	0.0137	0.0091	0.3685
HOM-HOM	0.0144	0.0102	0.0075	0.0669	0.0026	0.0292	0.0100	0.3742

LITERATURA

[1] Benjamin Baumer and Andrew Zimbalist. *The sabermetric revolution: Assessing the growth of analytics in baseball*. University of Pennsylvania Press, 2013.

[2] Igor Kononenko and Matjaž Kukar. *Machine learning and data mining: introduction to principles and algorithms*. Horwood Publishing, 2007.

[3] Robert P Schumaker, Osama K Solieman, and Hsinchun Chen. Predictive modeling for sports and gaming. In *Sports Data Mining*, pages 55–63. Springer, 2010.

[4] Arpad Elo. New USCF rating system. *Chess Life*, 16: 160–161, 1961.

[5] Lars Magnus Hvattum and Halvard Arntzen. Using ELO ratings for match result prediction in association football. *International Journal of forecasting*, 26(3): 460–470, 2010.

[6] Neil Paine. NLF Elo Ratings Are Back!, 2015. [Online; 10. 9. 2015]. Dostopno na: http://http://fivethirtyeight.com/datalab/nfl-elo-ratings-are-back/.

[7] Nate Silver and Robert Fischer-Baum. How We Calculate NBA Elo Ratings, 2015. [Online; 21. 5. 2015]. Dostopno na: http://fivethirtyeight.com/features/how-we-calculate-nba-elo-ratings/.

[8] Nate Silver. Lies, Damned Lies, 2006. [Online; 28. 6. 2006]. Dostopno na: http://http://www.baseballprospectus.com/article.php?articleid=5247.

[9] Mark E Glickman. The Glicko system. *Boston University*, 1995.

[10] Mark E Glickman. Example of the Glicko-2 system. *Boston University*, 2012.

[11] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill(TM): A Bayesian skill rating system. In *Advances in neural information processing systems*, pages 569–576, 2006.

[12] RJ Leake. A method for ranking teams: With an application to college football. *Management science in sports*, 4:27–46, 1976.

[13] Kenneth Massey. Statistical models applied to the rating of sports teams. *Bluefield College*, 1997.

[14] Wesley N Colley. Colley's bias free college football ranking method: The Colley matrix explained. *Princeton University, Princeton*, 2002.

[15] James P Keener. The Perron-Frobenius theorem and the ranking of football teams. *SIAM review*, 35(1): 80–93, 1993.

[16] Paul Kvam and Joel S Sokol. A logistic regression/Markov chain model for NCAA basketball. *Naval Research Logistics (NrL)*, 53(8):788–803, 2006.

[17] Mark Brown, Joel Sokol, et al. An improved LRMC method for NCAA basketball prediction. *Journal of Quantitative Analysis in Sports*, 6(3):1–23, 2010.

[18] Anjela Y Govan and Carl D Meyer. Ranking National Football League teams using Google's PageRank. In *AA Markov Anniversary Meeting*, 2006.

[19] Dean Oliver. *Basketball on paper: rules and tools for performance analysis*. Potomac Books, Inc., 2004.

[20] John Hollinger. *Pro Basketball Prospectus 2003-2004*. Brassey's, 2003.

[21] David J Berri. A simple measure of worker productivity in the National Basketball Association. *The business of sport*, 3:1–40, 2008.

[22] Justin Kubatko, Dean Oliver, Kevin Pelton, and Dan T Rosenbaum. A starting point for analyzing basketball statistics. *Journal of Quantitative Analysis in Sports*, 3 (3):1–22, 2007.

[23] Masaru Teramoto and Chad L Cross. Relative importance of performance factors in winning NBA games in regular season versus playoffs. *Journal of Quantitative Analysis in Sports*, 6(3), 2010.

[24] Tarek Baghal et al. Are the "Four Factors" Indicators of One Factor? An Application of Structural Equation Modeling Methodology to NBA Data in Prediction of Winning Percentage. *Journal of Quantitative Analysis in Sports*, 1(8):1559–0410, 2012.

[25] Erik Štrumbelj and Petar Vračar. Simulating a basketball match with a homogeneous Markov model and forecasting the outcome. *International Journal of Forecasting*, 28(2):532–542, 2012.

[26] Petar Vračar, Erik Štrumbelj, and Igor Kononenko. Modeling basketball play-by-play data. *Expert Systems with Applications*, 44:58–66, 2016.

[27] Daniel Cervone, Alex D'Amour, Luke Bornn, and Kirk Goldsberry. A multiresolution stochastic process model for predicting basketball possession outcomes. *Journal of the American Statistical Association*, pages 1–45, 2016.

[28] Alexander Franks, Andrew Miller, Luke Bornn, Kirk Goldsberry, et al. Characterizing the spatial structure of defensive skill in professional basketball. *The Annals of Applied Statistics*, 9(1):94–121, 2015.

[29] Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr, and Iain Matthews. Quality vs Quantity: Improved Shot Prediction in Soccer using Strategic Features from Spatiotemporal Data. In *Proc. 8th Annual MIT Sloan Sports Analytics Conference*, pages 1–9, 2014.

[30] Joachim Gudmundsson and Michael Horton. Spatio-Temporal Analysis of Team Sports–A Survey. *arXiv preprint arXiv:1602.06994*, 2016.

[31] Richard A Zuber, John M Gandar, and Benny D Bowers. Beating the spread: Testing the efficiency of the gambling market for National Football League games. *Journal of Political Economy*, 93(4):800–806, 1985.

[32] James D Dana and Michael M Knetter. Learning and efficiency in a gambling market. *Management Science*, 40(10):1317–1328, 1994.

[33] Mark E Glickman and Hal S Stern. A state-space model for National Football League scores. *Journal of the American Statistical Association*, 93(441):25–35, 1998.

[34] Shiladitya Sinha, Chris Dyer, Kevin Gimpel, and Noah A Smith. Predicting the NFL using Twitter. In *Proceedings of the ECML/PKDD 2013 Workshop on Machine Learning and Data Mining for Sports Analytics*, 2013.

[35] Rose D Baker and Ian G McHale. Forecasting exact scores in National Football League games. *International Journal of Forecasting*, 29(1):122–130, 2013.

[36] Keith Goldner. A Markov model of football: Using stochastic processes to model a football drive. *Journal of Quantitative Analysis in Sports*, 8(1), 2012.

[37] Michael J Maher. Modelling association football scores. *Statistica Neerlandica*, 36(3):109–118, 1982.

[38] Mark J Dixon and Stuart G Coles. Modelling association football scores and inefficiencies in the football betting market. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 46(2):265–280, 1997.

[39] Mark Dixon and Michael Robinson. A birth process model for association football matches. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47 (3):523–538, 1998.

[40] Havard Rue and Oyvind Salvesen. Prediction and retrospective analysis of soccer matches in a league. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 49(3):399–418, 2000.

[41] AC Titman, DA Costain, PG Ridall, and Kris Gregory. Joint modelling of goals and bookings in association football. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 178(3):659–683, 2015.

[42] Anthony C Constantinou, Norman E Fenton, and Martin Neil. pi-football: A Bayesian network model for forecasting Association Football match outcomes. *Knowledge-Based Systems*, 36:322–339, 2012.

[43] Tim Kuypers. Information and efficiency: an empirical study of a fixed odds betting market. *Applied Economics*, 32(11):1353–1363, 2000.

[44] John Goddard and Ioannis Asimakopoulos. Forecasting football results and the efficiency of fixed-odds betting. *Journal of Forecasting*, 23(1):51–66, 2004.

[45] Bryan L Boulier and Herman O Stekler. Are sports seedings good predictors?: An evaluation. *International Journal of Forecasting*, 15(1):83–91, 1999.

[46] David Forrest, John Goddard, and Robert Simmons. Odds-setters as forecasters: The case of English football. *International Journal of Forecasting*, 21(3):551 – 564, 2005.

[47] ChiUng Song, Bryan L. Boulier, and Herman O. Stekler. The comparative accuracy of judgmental and model forecasts of American football games. *International Journal of Forecasting*, 23(3):405 – 413, 2007.

[48] Martin Spann and Bernd Skiera. Sports forecasting: a comparison of the forecast accuracy of prediction markets, betting odds and tipsters. *Journal of Forecasting*, 28(1):55–72, 2009.

[49] Hal S Stern. A Brownian motion model for the progress of sports scores. *Journal of the American Statistical Association*, 89(427):1128–1134, 1994.

[50] Matt Goldman and Justin M Rao. Effort vs. concentration: the asymmetric impact of pressure on NBA performance. In *Proceedings MIT Sloan sports analytics conference*, pages 1–10, 2012.

[51] Alan Gabel, Sidney Redner, et al. Random walk picture of basketball scoring. *Journal of Quantitative Analysis in Sports*, 8(1):1416, 2012.

[52] Sears Merritt and Aaron Clauset. Scoring dynamics across professional team sports: tempo, balance and predictability. *EPJ Data Science*, 3(1):1, 2014.

[53] Kenny Shirley. A Markov model for basketball. In *New England Symposium for Statistics in Sports*, 2007.

[54] Tim Niblett and Ivan Bratko. Learning decision rules in noisy domains. In *Proceedings of Expert Systems' 86, the 6th Annual Technical Conference on Research and development in expert systems III*, pages 25–34. Cambridge University Press, 1987.

[55] Bojan Cestnik. Estimating probabilities: A crucial task in machine learning. In *European Conference on Artificial Intelligence 90*, pages 147–149, 1990.

[56] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[57] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[58] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[59] Igor Kononenko. On biases in estimating multi-valued attributes. In *Ijcai*, volume 95, pages 1034–1040, 1995.

[60] G. W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 75:1–3, 1950.

[61] E. S. Epstein. A scoring system for probability forecast of ranked categories. *Journal of Applied Meteorology*, 8: 985–987, 1969.

[62] Aleksandar Dimitriev and Erik Štrumbelj. Bayesian binary and ordinal regression with structured uncertainty in the inputs. In *Slovenian Conference on Artificial Intelligence : proceedings of the 19th International Multiconference Information Society - IS 2016*, volume A, pages 17–20. Institut Jožef Stefan, 2016.